integrata
cegos

Java Maintenance

Administration und Überwachung von Java-Prozessen

Literatur

- Offizielle Seiten und Whitepapers von Oracle
- Produkt-Dokumentationen
- Vorsicht:
  - Viel Literatur zum Thema „Administration“, ist stark programmierlastig.

Einige Hinweise

integrata cegos

- Dies ist kein Programmier-Seminar
  - Interessierte Teilnehmer können das Seminar aber gerne benutzten, die Quellcodes analysieren und erweitern
  - Fertige Anwendungen werden in elektronischer Form angeboten
- Die verwendeten Werkzeuge entstammen der Open Source Community
  - Kommerzieller Support wird von Herstellern zusätzlich angeboten.
- Dokumentation und Ressourcen stehen im Internet zur Verfügung
  - Beispiele unter
    `https://GitHub.com/Javacream/org.javacream.training.jvm`
    `https://GitHub.com/Javacream/org.javacream.training.tomcat`
- Konventionen
  - Befehle werden in `Courier-Schriftart` dargestellt
  - Dateinamen werden in *`kursiver Courier-Schriftart`* dargestellt
  - Links werden in `unterstrichener Courier-Schriftart` dargestellt

Copyright und Impressum

© Javacream

Javacream

Dr. Rainer Sawitzki

Alois-Gilg-Weg 6
81373 München

eMail: training@rainer-sawitzki.de

Alle Rechte, einschließlich derjenigen des auszugsweisen Abdrucks, der fotomechanischen und elektronischen Wiedergabe vorbehalten.

Inhalt

integrata cegos

1

EINFÜHRUNG

integrata cegos

1.1

JAVA AUF DEM SERVER

Java auf dem Server

integrata cegos

- Java-basierte Server-Lösungen sind seit 1998 im Einsatz und haben im Unternehmen eine enorme Verbreitung gefunden
- Die auf Java basierende Produkt-Palette ist dabei von erstaunlicher Breite:
  - Applikationsserver mit Transaktionsmonitor
  - Web Server
  - Content Management Systeme
  - Relationale Datenbanken
  - …
- Hintergrund für diese Flexibilität von Java ist, dass die eigentliche ausführende Instanz von Java-Anwendungen, die Java Virtual Machine (JVM), vereinfacht formuliert ein eigenes abstraktes Betriebssystem zur Verfügung stellt
  - Was innerhalb einer JVM installiert und ausgeführt wird ist erst einmal komplett egal.

1.2

DIE JAVA VIRTUAL MACHINE UND DAS BETRIEBSSYSTEM

Die Java Virtual Machine

integrata cegos

- Die Java Virtual Machine realisiert aber bei weitem nicht alle Aufgaben eines Betriebssystems, sie wird als Zusatz auf einem vorhandenen Plattform wie Linux oder Windows installiert
- Dann aber übernimmt die JVM die Speicherverwaltung, das Scheduling paralleler Prozesse (Threading), die Verwaltung von Ressourcen-Handles und ein Berechtigungskonzept
- Hardware-nahe Konfigurationen wie beispielsweise TCP/IP-Sockets, Dateisystem, Paging usw. bleiben aber weiterhin Aufgabe des Betriebssystems

JVM Spezifikation

- Die JVM genügt einer Spezifikation von Oracle, die auch auf unterschiedlichen Betriebssystemen einheitlich realisiert werden muss
  - Bestandteile dieser Spezifikation sind, wie eben erwähnt, Speicherverwaltung und Threads
  - Es sind aber zusätzlich eine ganze Reihe von Metriken definiert, die eine detaillierte Überwachung eines Java-Prozesses ermöglichen
  - Und diese Metriken stehen sofort identisch für alle Java-basierten Anwendungen einheitlich zur Verfügung
- Durch die Java Virtual Machine wird jedoch effektiv ein weiteres System eingeführt
  - Dieses bietet, positiv formuliert, neue Möglichkeiten für Optimierung und Tuning
  - Dazu muss jedoch, und das ist die Kehrseite, dieses System erst einmal beherrscht werden

Brauchen Administratoren Java-Kenntnisse?

integrata cegos

- Der „klassische“ Server-Administrator, der sein Betriebssystem und dessen Konfiguration in und auswendig beherrscht, kann von der Terminologie und den Möglichkeiten der JVM jedoch erst einmal abgeschreckt werden:
  - „Warum ist eine eigene Speicher-Konfiguration notwendig?“,
  - „Wieso braucht die JVM denn selbst ohne Last soviel Speicher?“,
  - „Was soll ich mit diesen Fehler-Ausgaben anfangen?“,
  - „Weshalb muss ich einen Connection-Pool zur Datenbank überwachen?“
- Weiterhin zeigt die Erfahrung aus der Praxis, dass zur wirklich effizienten Überwachung einer Server-seitigen Anwendung zumindest ein rudimentäres Wissen über typische Java-Anwendungen notwendig ist
  - Der Web-Auftritt eines Unternehmens verlangt eine andere Software und damit eine andere Konfiguration und die Überwachung anderer Metriken als ein Content Management System.

Notwendige Java-Kenntnisse

- Zum Glück beweist aber ebenfalls die Praxis, dass ein Großteil der Administration und Überwachung von Java-Anwendungen sich auf zwei allgemein gültige Bereiche zurückführen lässt:
  - Die allgemein verwendete Java Virtual Machine
  - Die Java-Applikationsserver wie beispielsweise Apache Tomcat oder Oracle Glassfish
- So unterscheiden sich völlig unterschiedliche Produkte wie beispielsweise ein Portalserver für den Web-Auftritt eines Unternehmens (z.B. Liferay) und ein Server für Web Services bezüglich Installation und Überwachung in erster Näherung nicht:
  - Beide sind beispielsweise auf einem Apache Tomcat installiert

Java Beispiele

integrata cegos

- Die in den folgenden Kapiteln beschriebenen Technologien können durch den Aufruf von Beispiel-Programmen nachvollzogen werden
- Diese Java-Anwendungen müssen nicht selbst programmiert oder erweitert werden
  - Der Quellcode steht jedoch selbstverständlich bei Bedarf zur Verfügung

integrata cegos

2

DIE VIRTUELLE MASCHINE

2.1

EINFÜHRUNG

Java Programme

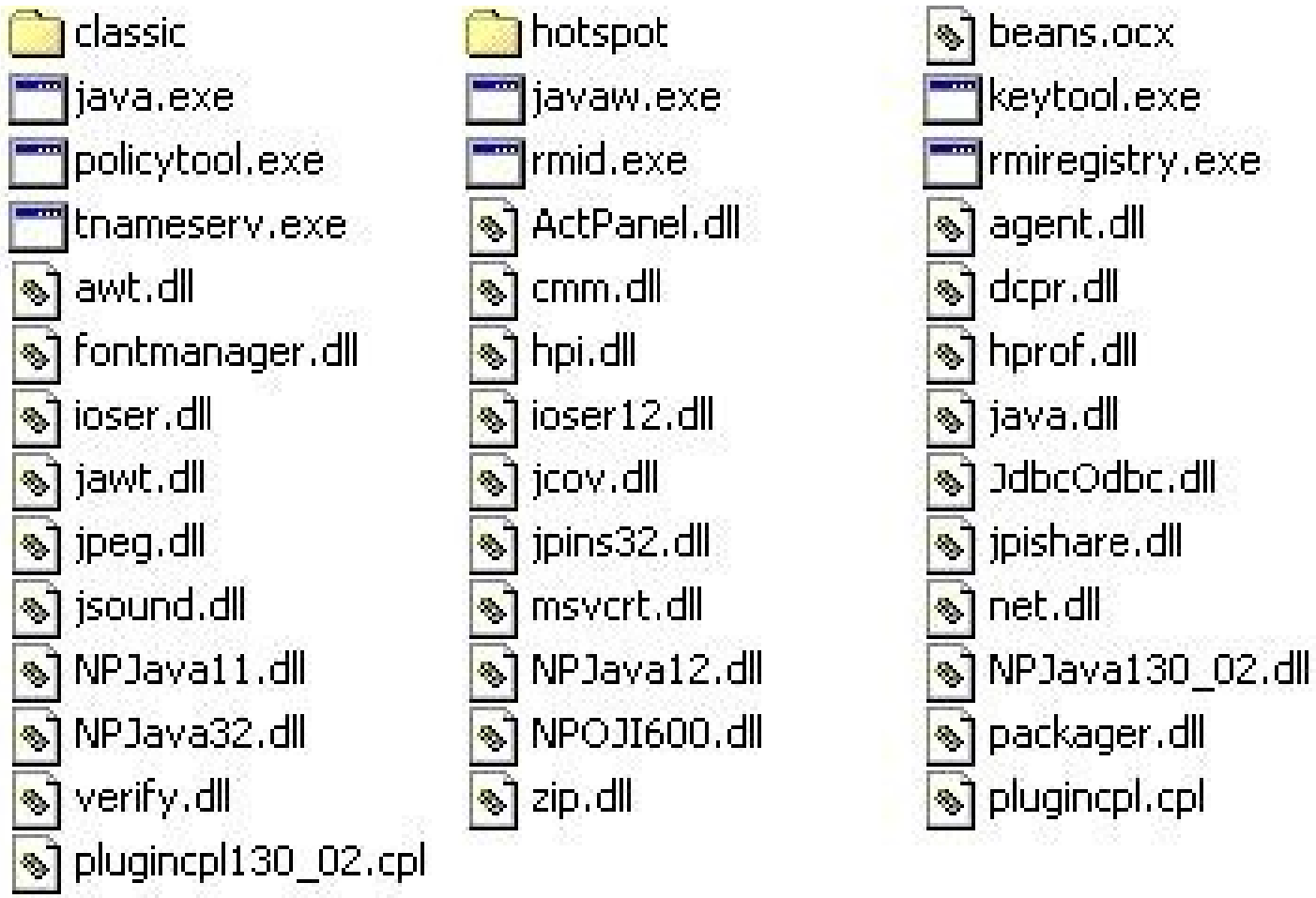
- Java Applikationsserver sind zumindest in großen Teilen in der Programmiersprache Java implementiert
- Für das Verständnis der Arbeitsweise eines Applikationsservers sowie erste elementare Administrationsaufgaben (nämlich die Administration des Java Prozesses selber) ist es notwendig, sich einen zumindest ein grundsätzliches Verständnis der Arbeitsweise einer Java Anwendung notwendig
- Ein Java-Programm wird nicht direkt auf dem Betriebssystem ausgeführt
  - Stattdessen wird ein eigener Prozess gestartet, die Java Virtuelle Maschine (JVM oder VM)
  - Die VM führt dann die eigentliche Java-Anwendung aus, die dadurch selber Plattform-unabhängig sind
  - Alle Hardware- und Betriebssystem-Spezialitäten werden von der VM komplett gekapselt

Java Installation

integrata cegos

- Für jede Plattformen, die Java unterstützen will, muss eine eigene Virtuelle Maschine zur Verfügung gestellt werden
- Die konkrete Implementierung erfolgt dabei natürlich durch plattformabhängige Programme

Java Binaries unter Windows



Java Distributionen

- Oracle spezifiziert die Virtuelle Maschine und stellt ein Verfahren zum Testen einer konkreten Implementierung zur Verfügung
  - Die Spezifikation ist frei einsehbar, z. B. über http://java.sun.com/docs/books/vmspec
  - Das heißt, Oracle bietet somit nicht nur ein Produkt „Virtual Machine“ an, sondern lizenziert und zertifiziert durch die JavaTest-Suite auch Anbieter, die die Spezifikation in eigenen Lösungen erfüllen
  - Dadurch konkurrieren bereits eine ganze Reihe Virtueller Maschinen für die unterschiedlichsten Plattformen, die jeweils bestimmte Aufgabenbereiche besonders effizient abdecken können

Java Provider

integrata cegos

- Die Produktpalette Virtueller Maschinen ist sehr umfassend
  - Es existieren spezielle Ausprägungen für Server, PC-Systeme und Kleingeräte, aber auch Maschinen, die zur Laufzeit die Ausführungsgeschwindigkeit der Programme optimieren können
- Oracles Virtuelle Maschine beinhaltet die „HotSpot“-Technologie zur Performance-Steigerung
  - Werkzeuge zum Testen von Java-Programmen benötigen spezielle Routinen zum automatischen Sammeln der Profiling-Information zur Ausführungszeit
- Der Administrator einer Java-Anwendung kann somit aus verschiedenen konkurrierenden Produkten das am besten geeignete wählen
  - Der Java-Programmierer sieht aber nur die allgemein gültige Schnittstelle der Spezifikation und kann sich auf Grund der Zertifizierung von Oracle auf eine korrekte Implementierung der Spezifikation verlassen

2.2

JAVA PROGRAMMIERUNG

Java Programme

- Jede Java-Anwendung ist in relativ kleine Module unterteilt, die so genannten Klassen
  - Diese Klassen haben Standardmäßig die Endung „.class“
- Nachdem ein Applikationsserver ein komplexe Java-Anwendung ist und zusätzlich noch die Klassen der Anwendungen kommen, die innerhalb des Servers installiert sind, kann die Virtuelle Maschine auch gezippte Verzeichnisbäume lesen
  - Diese Dateien, die Java-Archive haben die Endung „.jar“ (Englisch für Gefäß) und enthalten beliebig viele Klassen
- Die Virtuelle Maschine kann nun nicht nur ein Java-Archiv lesen sondern wiederum eine beliebige Menge davon

Voll qualifizierter Klassenname

integrata cegos

- Die hierarchische Paketstruktur erfordert zur eindeutigen Identifikation einer Java-Klasse nicht nur den Namen der Klasse, sondern auch die Angabe des Paketes
- Dies erfolgt durch den voll qualifizierten Klassennamen, der den Punkt als Trennzeichen verwendet
- Konventionen:
  - Die Systemklassen befinden sich entweder im Paket java oder in javax
  - Paketnamen bestehen aus Kleinbuchstaben
  - Klassennamen beginnen mit einem Großbuchstaben
  - Alle anderen Klassen sollen in eine Paketstruktur integriert sein, deren beide Hauptpakete mit der „umgedrehten“ WWW-Adresse beginnen

Bibliotheken der Java-Installation

integrata cegos

- Die Virtuelle Maschine lädt nur bestimmte Klassen direkt in den Arbeitsspeicher
- Diese so genannten „Bootstrap-Klassen“ sind die Systemklassen, die in der Datei rt.jar im lib-Verzeichnis gefunden werden
- Ein weiteres Java-Archiv, i18n.jar enthält ebenfalls Systemklassen
- Eine besondere Systemklasse ist der so genannte Klassenlader
  - Dieser ist in der Lage, eine frei wählbare Anzahl von Lokationen nach Java-Archiven und Java-Klassen zu durchsuchen und der Virtuellen Maschine zur Verfügung zu stellen
  - Um die Anzahl und Ort der zu durchsuchenden Quellen frei einstellen zu können, verwendet der Standard-Klassenlader die Umgebungsvariable CLASSPATH

Der Klassenpfad

- Dieser Pfad enthält eine über Semikolon (Windows) oder Doppelpunkt (Solaris) getrennte Liste von Java-Archivdateien und Verzeichnissen
- Die Virtuelle Maschine versucht nun, eine in einem Programm verwendete Klasse dadurch zu laden, indem
  - als erstes die Systemklassen durchsucht werden und, falls die angesprochene Klasse darin nicht gefunden wird,
  - danach über den ClassLoader der Klassenpfad in der angegebenen Reihenfolge
- Wird die benötigte Klasse gefunden, wird sie einmalig in den Speicher geladen
  - Sonst tritt ein Fehler auf
- Der Klassenpfad kann sowohl als Umgebungsvariable gesetzt werden als auch der Virtuellen Maschine beim Programmstart als Option (-classpath bzw. -cp) mitgegeben werden

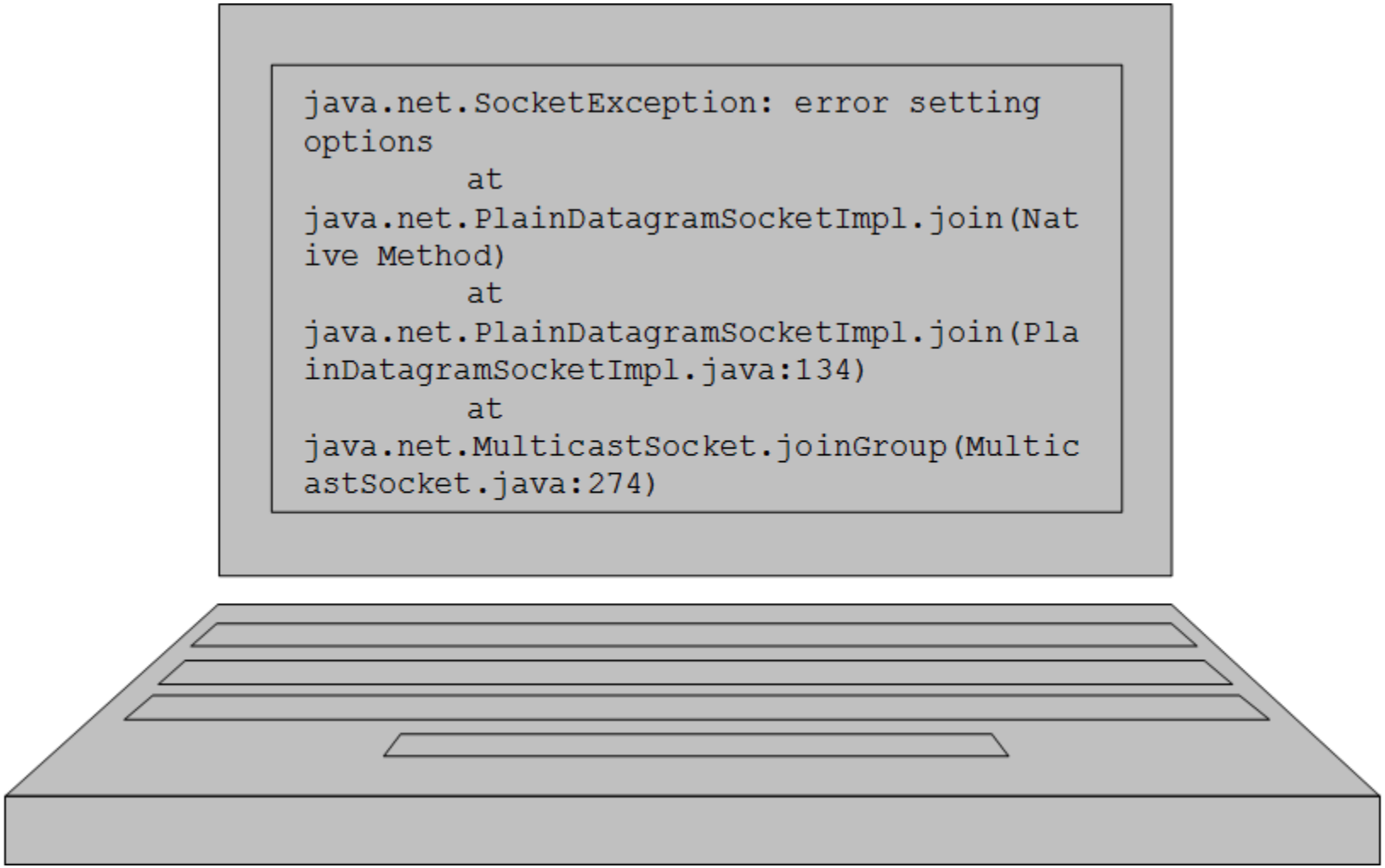
2.3

FEHLERPROTOKOLLE

Stack Traces

- Tritt während eines Programms eine Fehlersituation auf, so wird von der Anwendung ein „Stack Trace“ geschrieben
- Diese Stack Traces haben in Java-Anwendungen einen bestimmten Aufbau:
  - Der Fehlertyp wird durch eine Java-Klasse, die „Exception“, definiert, deren voll qualifizierter Klassenname den Stack Trace einleitet
  - Es folgt, falls vorhanden, ein mehr oder weniger sprechender Fehlerbeschreibungstext
  - Als drittes erfolgt der eigentliche Stack Trace, der den gesamten Aufruf, also den Method Frame, ausgibt
  - Ist die Anwendung auf spezielle Art und Weise erstellt worden, werden hierbei sogar noch die Zeilennummern des Quellcodes mit ausgegeben
    - Für die Fehlersuche für den Entwickler eine unschätzbare Hilfe

Ein Stack Trace



Interpretation des Stack Traces

integrata cegos

- Die Interpretation des Stack Traces ist in der Praxis häufig jedoch nicht so einfach:
  - Fehlende beschreibende Texte
  - Vielfache Ausgabe von Fehlermeldungen, da der Stack Trace im Rahmen des Programms an mehreren Stellen ausgegeben wird
  - Schwer zu lesende Ausgaben durch verschachtelte Exceptions
  - Bei verteilten Anwendungen ist die Frage, auf welcher Maschine der eigentliche Fehler protokolliert wurde
    - Eine unspezifische „java.rmi.RemoteException“ auf Client-Seite kann beispielsweise auf dem Server als „java.sql.SQLException: Duplicate Key“ eine klare Ursache haben
- Insgesamt erfordert die Interpretation von Stack Traces einiges an Erfahrung
  - Wichtig für den Administrator eines Systems ist es dabei, Stack Traces möglichst vollständig zu protokollieren und damit aussagekräftig an den Entwickler weiterleiten zu können

integrata cegos

2.4

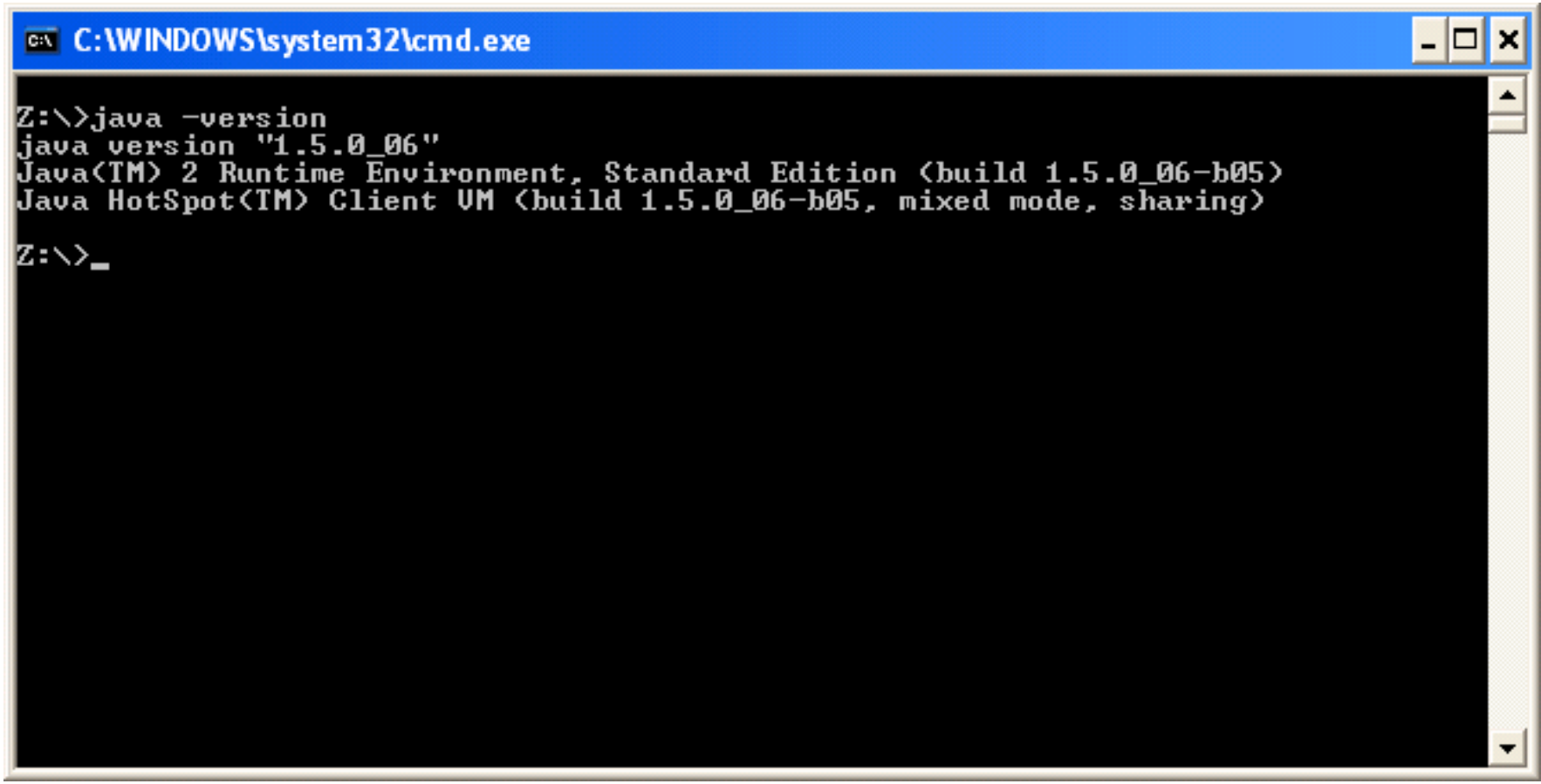
AUFRUF EINES JAVA-PROGRAMMS

Aufruf eines Java-Programms

- Der Aufruf einer Java-Anwendung erfolgt durch den Aufruf des ausführbaren Java-Prozesses namens „java“
  - Falls dieses Programm nicht im Pfad für ausführbare Dateien des Betriebssystems gefunden wird, kann der Aufruf auch mit vollständiger Pfadangabe erfolgen
- Um die Installation der Java-Laufzeitumgebung zu testen, kann der Aufruf mit der Option „-version“ erfolgen

Java Versionen

integrata cegos



Fehlermeldung bei falscher Java-Version

- Nachdem auf einer Rechner-Installation auch mehrere Java Laufzeitumgebungen in verschiedenen Versionen parallel installiert sein können gehört dieser Test beim Auftreten „merkwürdiger“ Fehler zum Standard-Fehlersuche
- So zeigt die folgende Fehlermeldung an, dass eine Java-Anwendung von einer Virtuellen Maschine ausgeführt werden soll, die eine zu niedrige Version aufweist:
- Exception in thread "main" java.lang.UnsupportedClassVersionError: org/javacream/books/client/BooksClient (Unsupported major.minor version 49.0)

Classpath

- Der Klassenpfad, in dem alle Klassen gefunden werden, die die Anwendung ausmachen, wird durch die „classpath“-Option gesetzt
- Auch hier können natürlich Fehler auftreten:
  - `java.lang.ClassNotFoundExeption <Klassenname>`
  - `java.lang.NoClassDefFoundError <Klassenname>`
  - `java.lang.LinkageError <Klassename>`
- Die ersten beiden Exception-Ausgaben sind relativ einfach zu interpretieren
  - Es wird eine Klasse benötigt, die im gesamten Klassenpfad nicht zu finden ist
  - Der Unterschied ist nur für einen Java-Programmierer relevant

Verschiedene Klassenversionen

- Die dritte Fehlermeldung ist hingegen recht kritisch:
  - Sie zeigt an, dass die Anwendung eine Klasse sehr wohl findet, nun aber mehrfach in unterschiedlichen Versionen
  - Ein Applikationsserver benutzt viele verschiedene Klassenlader und damit kann es sehr wohl zu dieser Fehlersituation kommen
- Die Behebung des Fehlers kann aller Erfahrung nach jedoch nicht vom Administrator des Applikationsservers alleine übernommen werden
  - Der Fehler liegt mit höchster Wahrscheinlichkeit an den zu installierenden Anwendungsprogrammen bzw. den erzeugten Java-Archiven

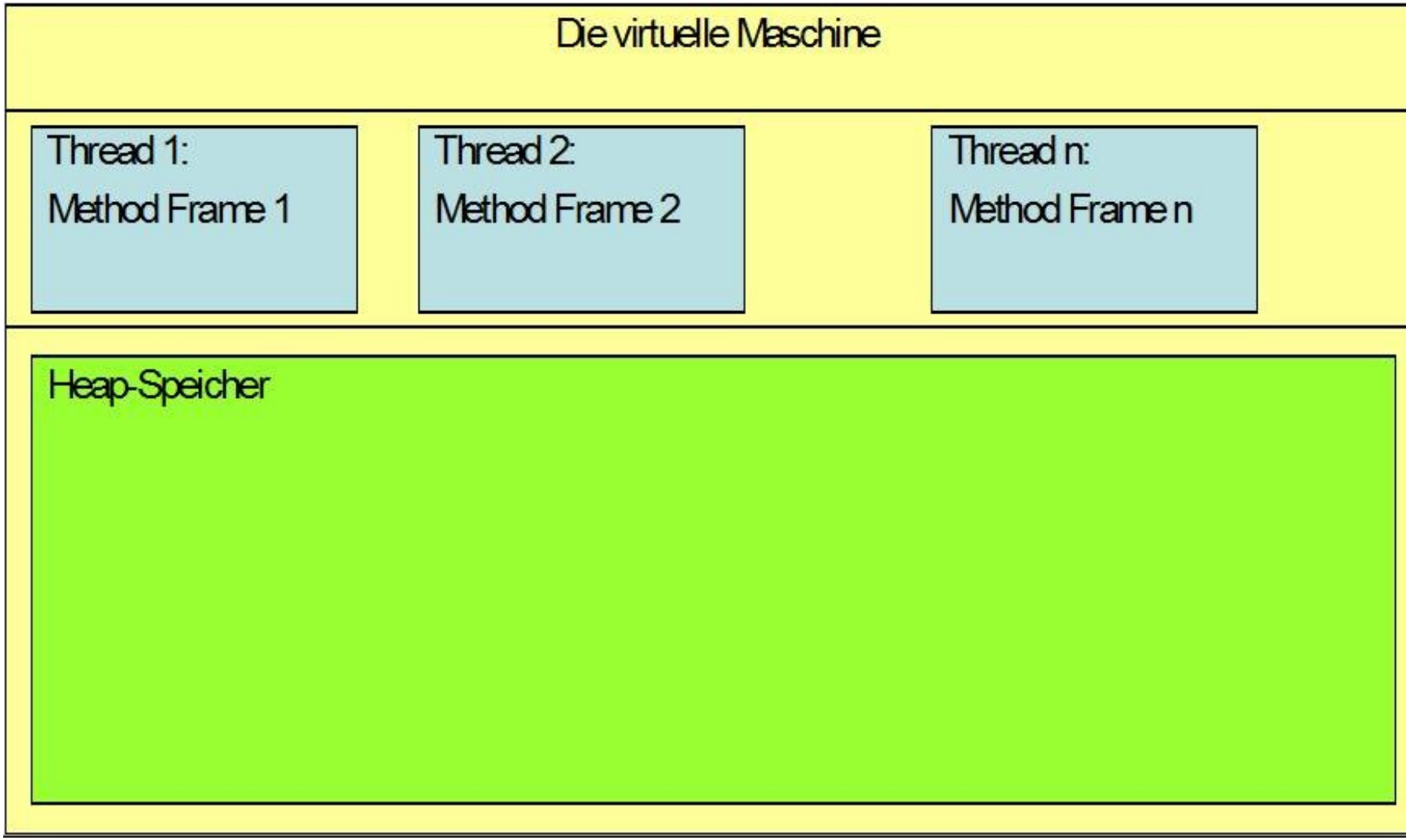
2.5

THREADS

Technik

- Die Virtuelle Maschine führt nun ihre Aufgaben nicht in einem einzigen Prozess seriell durch sondern verwendet dafür verschiedene Threads
- Viele (auf Server-Seite sind mehrere Hundert Threads keine Seltenheit) Threads laufen innerhalb der Virtuellen Maschine „parallel“ und können so beispielsweise auf einer Mehrprozessormaschine tatsächlich nebenläufig ausgeführt werden
- Jeder dieser Threads hat einen eigenen Stack-Bereich, den so genannten „Method Frame“ für seine aktuellen Methodenaufrufe, lokalen Parameter etc.

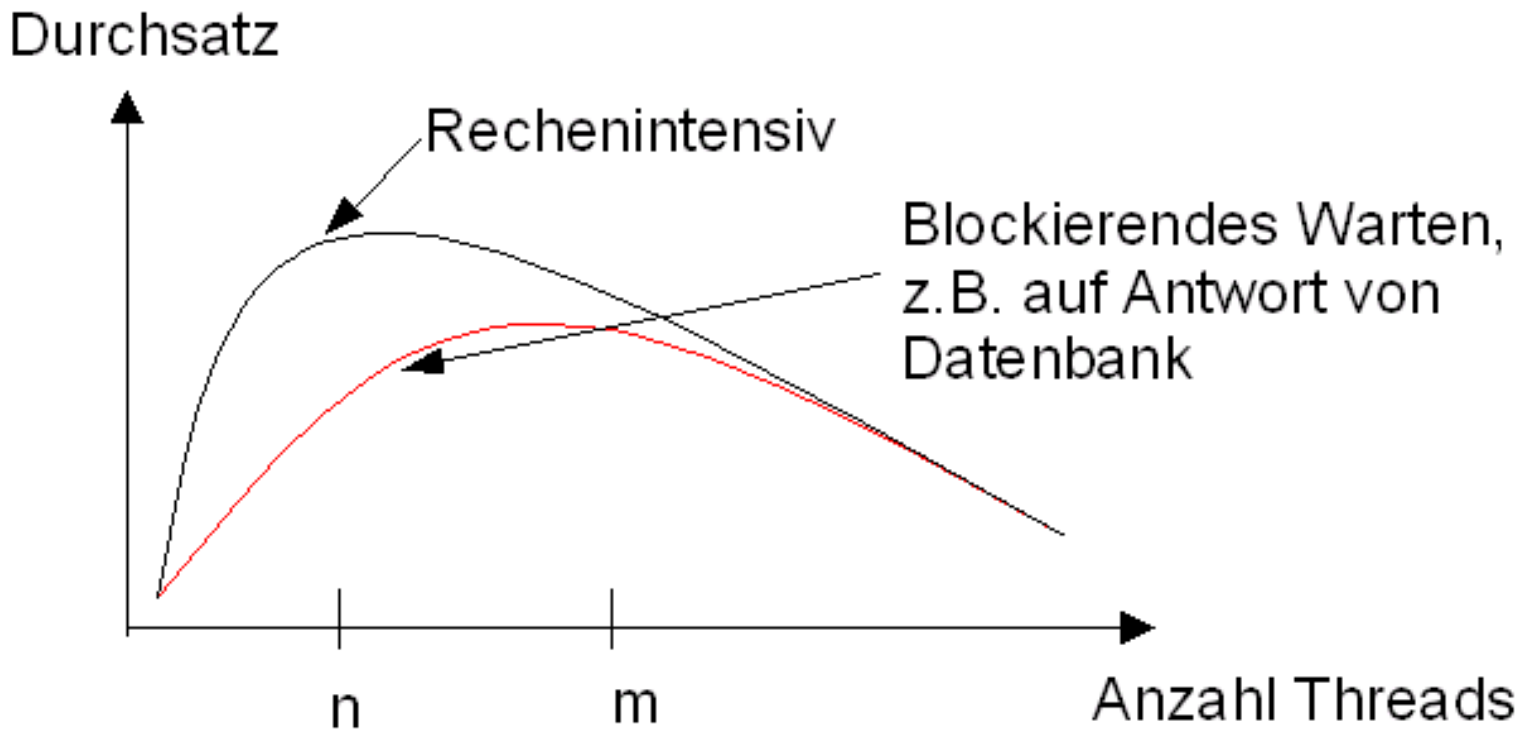
JVM und Threads



Optimierung der Threads

- Threads sind für eine Virtuelle Maschine relativ aufwändig zu verwalten
  - Die Menge an Threads, die innerhalb eines Java-Prozesses effizient parallel ablaufen können, hängt natürlich stark von der verwendeten Hardware ab
- Daneben gibt es aber auch einige Regeln und Kriterien, die für die optimale Einstellung relevant sind
  - So verwaltet ja wie bereits erwähnt ein Applikationsserver einen oder mehrere Thread Pools um eingehende Requests bearbeiten zu können

Optimierung des Durchsatzes



Erläuterung

integrata cegos

- Bei einer rein Rechenintensiven Anwendung, die ausschließlich auf dem Applikationsserver läuft ist der optimale Durchsatz schnell bestimmt:
  - Nachdem jeder Thread-Wechsel selber wiederum Prozessorleistung benötigt liegt das Maximum bei der Anzahl der Prozessoren
- Nicht so einfach (und leider auch eher die Realität) sind Anwendungen, die zumindest zeitweise auf Antwort eines anderen Systems warten und während dieser Wartezeit den aufrufenden Thread innerhalb des Applikationsservers blockieren
  - Nun kann die Menge der konfigurierten Threads selbstverständlich die Anzahl der Prozessoren deutlich übersteigen
  - Der Optimalwert m ist die Anzahl von Threads, bei der zum ersten mal 100% Auslastung erreicht werden
    - Der Umkehrschluss „100% Prozessorlast = Optimale Konfiguration“ ist allerdings falsch: Ein zu groß dimensionierter Thread Pool führt durch überflüssige Threadwechsel ebenfalls zur Auslastung der Prozessoren, ohne dass Anfragen abgearbeitet werden

2.6

HEAP-SPEICHER

- Der Prozess „Virtuelle Maschine“ verwaltet den ihm zugeteilten Speicher und Prozessorzeit in einer eigenen, wohldurchdachten Art und Weise
- Der Speicher der virtuellen Maschine heißt „Heap“ und entspricht im Wesentlichen dem zugeteilten Hauptspeicher
- Darin werden Informationen in Form von Objekten (in etwa frei definierbaren Strukturen) und Arrays (Listen von Objekten und Datentypen) abgelegt
- Die Größe des Heap-Speichers wird beim Programmstart der Virtuellen Maschine über die Optionen
  - -Xmxn
    - Maximaler Heap-Speicher
  - -Xmsn
    - Startgröße des Heap-Speichers

Hinweise

- Die aktuelle Größe des Heap-Speichers und die reale Größe des VM-Prozesses können sich unterscheiden, da die Größe zur Laufzeit intern angepasst werden kann ohne frei gewordenen Speicher sofort dem Betriebssystem zurück zu geben
  - Werkzeuge, die einfach die für den Prozess „Virtuelle Maschine“ reservierten Speicher messen, sind für Monitoring und Analyse deshalb vollkommen ungeeignet
- Auch ist die aktuelle Größe des Heap-Speichers nur bedingt ein Kriterium für den aktuellen Speicherbedarf einer installierten Anwendung:
  - Wird ein Server beispielsweise gestartet mit java –Xms256m –Xmx256m so ist der Heap immer 256 Megabyte groß, obwohl vielleicht noch gar keine Anwendung installiert wurde und deshalb „eigentlich“ kein Speicherbedarf vorhanden ist

OutOfMemoryError

integrata cegos

- Benötigt ein Java-Prozess mehr Speicher als ihm beim Aufruf zugestanden wurde, wird ein java.lang.OutOfMemoryError geworfen
- Das Auftreten dieses Fehlers führt jedoch nicht zu einer Beendigung der Virtuellen Maschine:
  - Kann die VM nach Auftreten des Fehlers wieder Speicher freigeben, so arbeitet sie ganz normal weiter
  - So kann beispielsweise eine Datenbankabfrage mit ungeschickten Selektionskriterien eine Unmenge von Datensätzen liefern, die den Speicher des Applikationsservers sprengt
  - Dann wird das weitere Einlesen der Daten abgebrochen, aber es werden auch die bereits gelesenen Daten verworfen und damit ist wieder Speicher frei

2.7

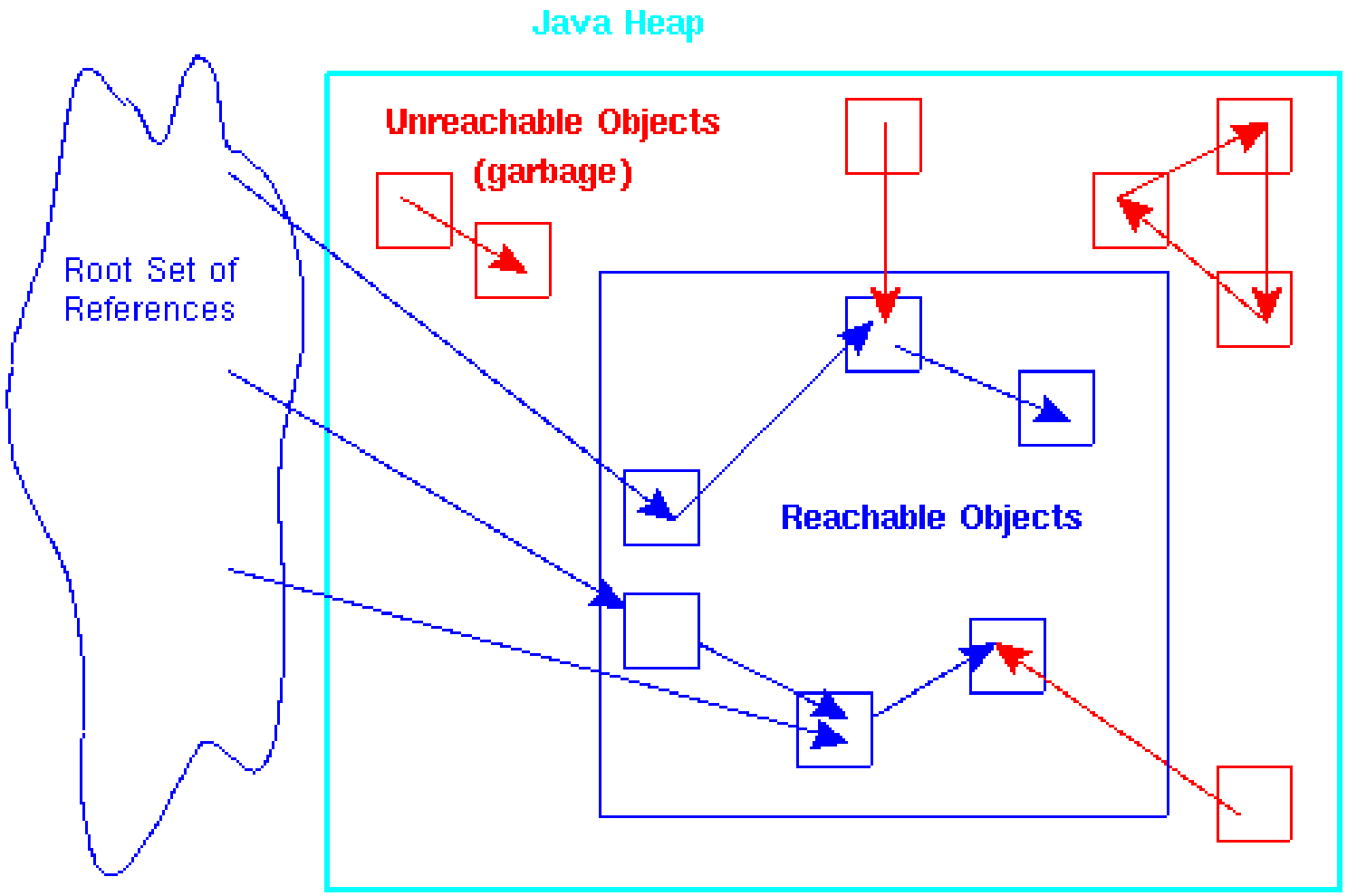
GARBAGE COLLECTION

Motivation der Garbage Collection

integrata cegos

- Ein, insbesondere für Java-Server wichtiges Feature der Virtuellen Maschine ist die Garbage Collection, die automatisch den Heap von nicht mehr zugreifbaren und damit unnötigen Objekten befreit
- Dies verhindert eine Vielzahl möglicher Programmierfehler, die zum Auftreten von Speicher-Lecks führen könnten

Prinzipielle Arbeitsweise



Wichtigkeit der Garabage Collection

integrata cegos

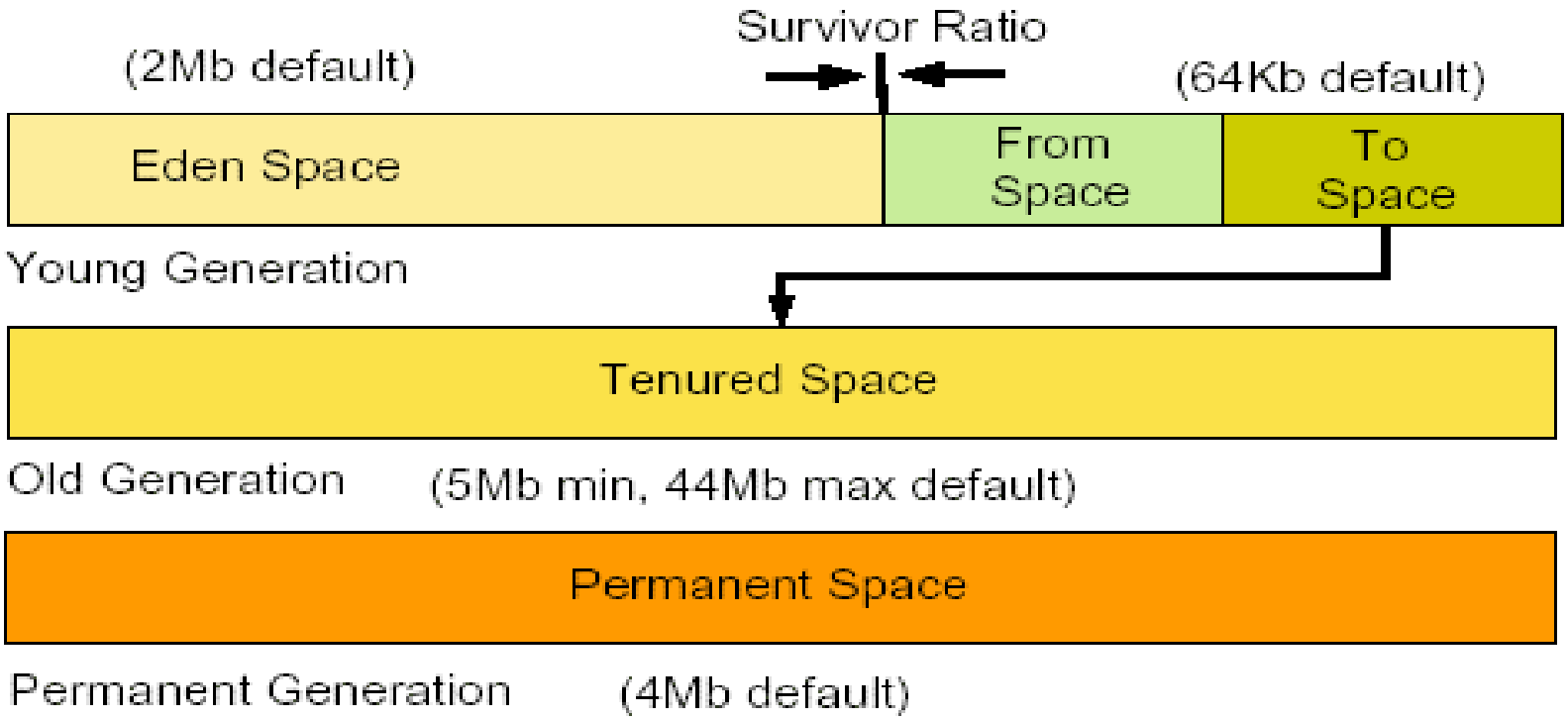
- Die korrekte Konfiguration der Garbage Collection ist insbesondere für lang laufende Java-Anwendungen im Hochlastbereich einer Server-Anwendung kritisch
- Die modernen Java-Laufzeitumgebungen bieten dafür eine Vielzahl von Optionen, die im Folgenden detailliert erläutert werden

2.8

STRUKTUR DES HEAP-SPEICHERS

Der Heap Speicher der JVM

integrata cegos



Bereiche

- Young Generation
- Eden Space
- From Space
- To Space
- Old Generation
- Permanent Generation
  - Ab Java 8 nicht mehr im Einsatz

Der Eden Space

- Alle neu angelegten Objekte werden im Eden Space angelegt und verbleiben dort bis zur ersten Garbage Collection

Der Survivor Space

integrata cegos

- Zwischenspeicher für Objekte, die noch gültig sind aber noch nicht alt genug für die Old Generation sind
- Objekte im Eden-Space, die von der Garbage Collection nicht entfernt werden dürfen, weil sie noch benötigt werden, werden alternierend in den jeweils nächsten Survivor Space kopiert
- Nach einer Garbage Collection in der Young Generation ist damit der Eden Space sowie ein Survivor Space leer, der andere Survivor Space hält die noch gültigen Objekte

Die Old/Tenured Generation

- Länger gültige Objekte werden aus dem Survivor Space in die Tenured Generation transferiert
- Auch in der Tenured Generation läuft eine Garbage Collection ab, die diese bereinigen kann

Die Permanent Generation

integrata cegos

- Klassenobjekte werden in der Permanent Generation abgelegt, da diese Objekte in der Regel niemals entladen werden
- Für die Permanent Generation wird prinzipiell überhaupt keine Garbage Collection benötigt
  - Trotzdem wird auch dieser Bereich bereinigt
  - Die Permanent Generation war eine experimentelle Idee der JVMs vor Java 8, die verworfen wurde

Zusammenfassung

integrata cegos

- Neu angelegte Objekte werden im Eden Space erzeugt
- Ein Survivor Space ist stets leer
- Der zweite wird während einer „Copy Collection“ mit den gültigen Objekte aus Eden und dem ersten befüllt
- Objekte, die genügend alt sind, werden in die Old Generation verschoben
- Der Trick bei der Konfiguration der Garbage Collection ist nun die, die Größe der Bereiche den realen Anwendungen anzupassen und somit ein „rundes“ laufen des Collectors zu garantieren

Speicher ohne Garbage Collection

integrata cegos

- Gibt es tatsächlich auch: Native Memory
  - Eine Art RAM-Disk
- Dafür gibt es aber noch kein "offizielles" Programmiermodell
  - Allerdings wird dies in der nächsten Sprachversion erfolgen
- Moderne Java-basierte Cache-Produkte benutzen bereits Native Memory
  - z.B. EHCache/Terracotta

Profiling der Garbage Collection

integrata cegos

- Profiling der Anwendung bezüglich des Speicherverhaltens ist insbesondere für Server-Anwendungen essenziell
- Bei Fehlkonfigurationen sind Pausenzeiten von mehreren Sekunden („Stop the World Collections“) beziehungsweise drastische Performance-Einbußen die Regel

Aufrufoptionen

integrata cegos

- Mit den folgenden Einstellungen liefert die Java Virtual Machine detaillierte Informationen über das Laufzeitverhalten des Garbage Collectors:
  - `-verbose:gc`
  - `-Xrunhprof`
  - `-XX:+PrintGCDetails`
  - `-XX:+PrintGCTimeStamps`
  - `-XX:+PrintHeapAtGC`

Profiling-Werkzeuge

integrata cegos

- Es existieren aber auch eine Reihe von Werkzeugen, die die Ergebnisse in analysierbarer Form aufbereiten:
  - Eine zwar alte aber sehr schöne Oberfläche ist der GCViewer, ein Open Source Werkzeug der Tagtraum Industries
  - Als Bestandteil der Java-Installation VisualVM

2.9

SPEICHERKONFIGURATION

Optionen

- Berechnung der Größen:
  - Eden = NewSize – ((NewSize / (SurvivorRatio + 2)) * 2)
  - From space = (NewSize – Eden) / 2
  - To space = (NewSize – Eden) / 2
- Einstellung der New Generation:
  - `-XX:NewSize`
  - `-XX:MaxNewSize`
  - `-XX:SurvivorRatio`
  - `-XX:MaxTenuringThreshhold`

Speicheroptionen

- Allgemeine Einstellungen
  - `-Xms`
  - `-Xmx`
- Einstellung der Old Generation:
  - `-XX:MinHeapFreeRatio`
  - `-XX:MaxHeapFreeRatio`
- Einstellungen für die Permanent Generation
  - `-XX:PermSize`
  - `-XX:MaxPermSize`
  - `-Xnoclassgc`

Einfluss der Parameter auf die Garbage Collection

integrata cegos

- Die Garbage Collection muss nicht mehr benötigte Objekte schnell und effizient aus dem Heap-Speicher entfernen können
- Insbesondere bei großen Heap-Speichern kann die Garbage Collection erheblich Ausführungsgeschwindigkeit kosten
- Dieser Einfluss zeigt sich besonders bei Mehrprozessor-Systemen, da nicht alle Phasen der Garbage Collection parallel laufen können

Major/Minor Collections

- Minor Collections ohne Defragmentierung
- Major Collections mit Defragmentierung

Kategorisierung der Collectors

- Copying Collection
- Mark-Compact
- Inkrementeller Collector
- Parallel Copying Collector
- Concurrent Collector
- Parallel Scavenge Collector

integrata cegos

2.10

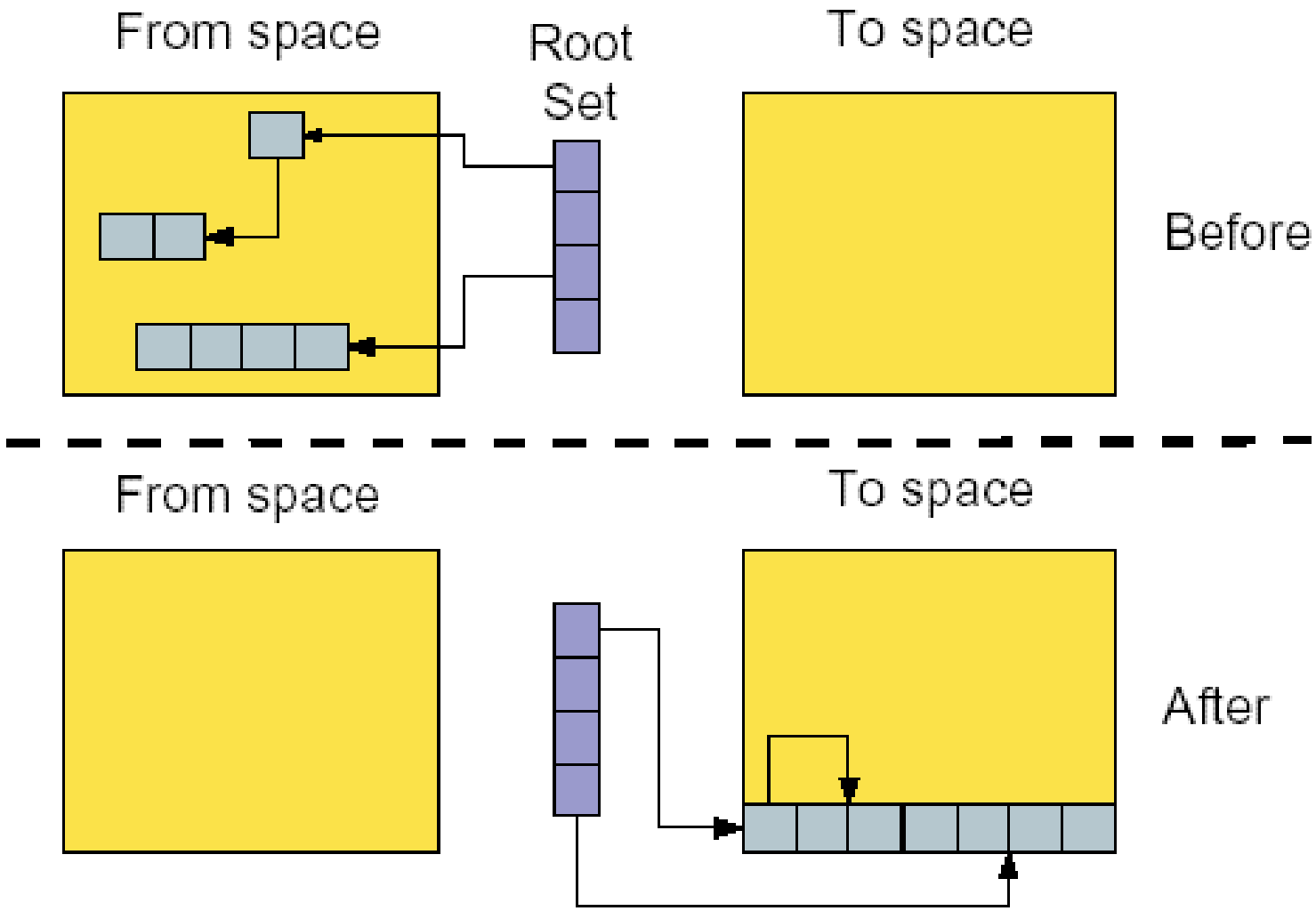
COLLECTOR-IMPLEMENTIERUNGEN

Copying/Scavenge Collector

- Alle Objekte werden aus dem Heap-Speicher entfernt
- Die „Lücken“ bleiben erhalten, um neue Objekte schnell erzeugen („wieder beleben“) zu können
- Gültige Objekte werden in einen anderen Speicherbereich kopiert
- Dieses Verfahren ist sehr schnell, benötigt aber stets leeren Speicher
- Die Young Generation ist genau auf diesen Typ der Collection hin ausgerichtet, die Copying Collection ist das Verfahren für Minor Collections

Arbeitsweise des Copying Collectors

integrata cegos



Eigenschaften

- Ausgeprägter „Stop the World“-Effekt bei Defragmentierung
- Sehr effizient und relativ einfach
- Pause durch die Garbage Collection ist direkt proportional zur Gesamtgröße der Live-Objekte, die kopiert werden müssen
- Hier kommt die Survivor Ratio zum Tragen

Mark sweep/compact Collector

integrata cegos

- Hier werden ungültige Objekte aus dem Speicher entfernt
- Compact: Anschließend werden die entstandenen Lücken im Speicher eliminiert
- Arbeitsweise
  - Exakte Nachverfolgung der Objektgrafen
  - Markierung aller Objekte, die erreichbar sind
- Problem
  - Unterschiedlich große Objekte fragmentieren den Speicher
  - Dauer der Collection Pause proportional zur Größe des Heaps insgesamt, nicht der Menge von Objekten

Incremental Collector

- Die Garbage Collection läuft ständig in einem Hintergrundthread
- Damit werden längere Pausen („Stop the world“-Collections) vermieden
- Geeignet für
  - Hoch-verfügbare Server-Applikationen
  - Benutzung großer Datenstrukturen
  - (Spiele, Animationen...)

Parallel Copy Collector (JDK 1.4.1)

integrata cegos

- Ähnlich dem Copy Collector
- Immer noch „Stop the World“
- Aber: Pro Prozessor wird ein eigener Thread verwendet
- Einstellungen:
  - -XX:+UseParNewGC
- Deaktivieren bei Einprozessor-Maschine
  - XX:ParallelGCThreads=<num>
- Standard: Anzahl der CPUs
- Erzwingt parallele Collection auf Einprozessor-Maschine

Weitere Einstellungen

- Concurrent GC
  - -XX:+UseConcMarkSweepGC
- Parallel Scavenge Collector
  - Ähnlich dem Parallel Copy Collector
  - „Stop the world“
  - Jedoch besser geeignet für große Young Generations (12 – 80 GB)
- Einstellungen:
  - -XX:+UseParallelGC
  - -XX:ParallelGCThreads=<num>
  - -XX:+UseAdaptiveSizePolicy (Automatische Optimierung der Größe der Young Size)

Auswahl-Kriterien

integrata cegos

- Entscheidend dafür ist die Lebensdauer der verwendeten Objekte:
- Viele kurzlebige, „kleine“ Objekte oder langlebige Datenstrukturen?
- Ist ein fragmentierter Speicher ein Problem?
- Sind Pausen durch die Garbage Collection tolerierbar?
- Interpretation der Grafik:
- Die allermeisten Objekte überstehen keine einzige Minor Collection
- Die Lebensdauer dieser Objekte ist nicht wesentlich größer als ein Methodenaufruf
- Ein eher geringer Teil der Objekte wird bei Major Collections entfernt
- Langlebige Objekte sind häufig „unsterblich“

integrata cegos

2.11

TYPISCHE BETRIEBSSITUATIONEN

Einführung

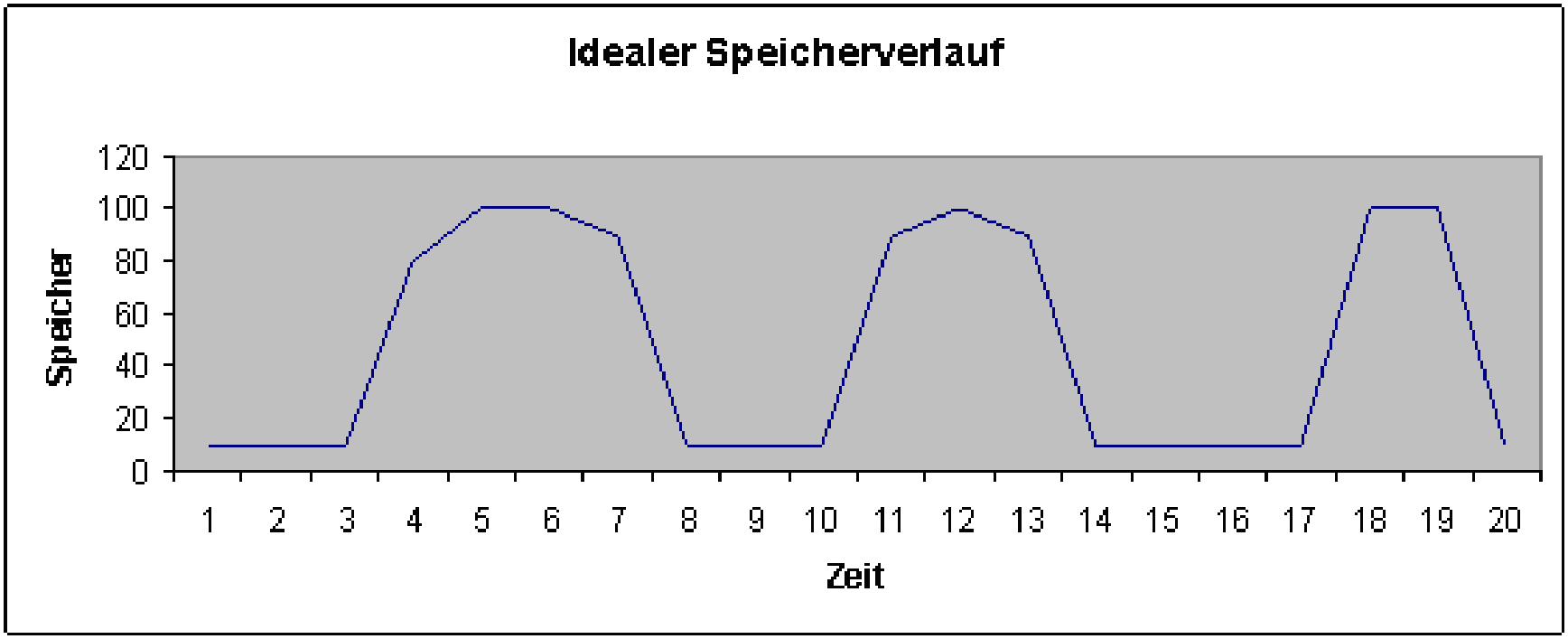
integrata cegos

- Die folgenden Abschnitte beschreiben eine Auswahl typischer Betriebssituationen einer lang laufenden virtuellen Maschine
- Diese Situationen sind natürlich idealisiert und stellen sich in der Realität wesentlich unklarer dar
  - Insbesondere zeigt sich bei allen lang laufenden Java-Anwendungen auch ohne Last ein langsames Anwachsen des Speicherbedarfs, der durch eine Full Garbage Collection bereinigt werden muss
- Nichtsdestotrotz können die hier beschriebenen Szenarien mit etwas Erfahrung erkannt und für die Analyse eines Problems sinnvoll eingesetzt werden

Idealer Speicherverlauf

- Im Idealfall finden sich alle Requests, die vom Java-Server bearbeitet werden, als Peaks im Speicherverlauf wieder

Heap-Speicher bei idealem Speicherverlauf



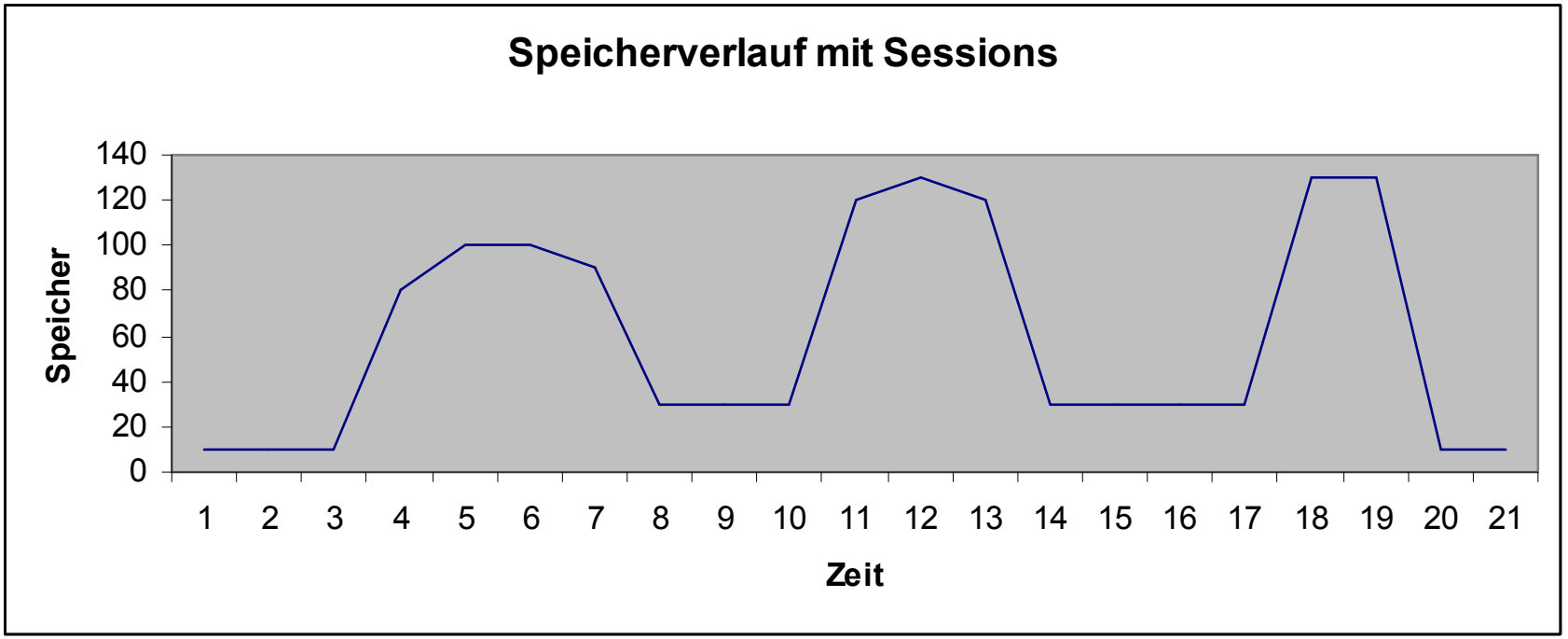
Erläuterung

- Die Breite des Peaks hängt davon ab, wie lange die Request-Verarbeitung dauert
- Nach dem Ende des Requests wird hier exakt der ursprüngliche Wert des Speicherverbrauchs erreicht
  - Alle notwendigen Objekte können von der Garbage Collection entfernt werden
  - Bevorzugt sollten dies Copy-Collections innerhalb der New Generation sein

Langlebige Sessions

- Hier werden durch Requests auf dem Server zusätzlich Sessions (Login eines Anwenders) aufgebaut, die zu breiten Plateaus führen
  - Erst beim Beenden der Session (Logout) wird der Speicher wieder komplett bereinigt
- Durch die aus Sicht der Virtuellen Maschine lang laufenden Sessions wird hier mit Sicherheit die Tenured Generation genutzt werden

Langlebige Sessions

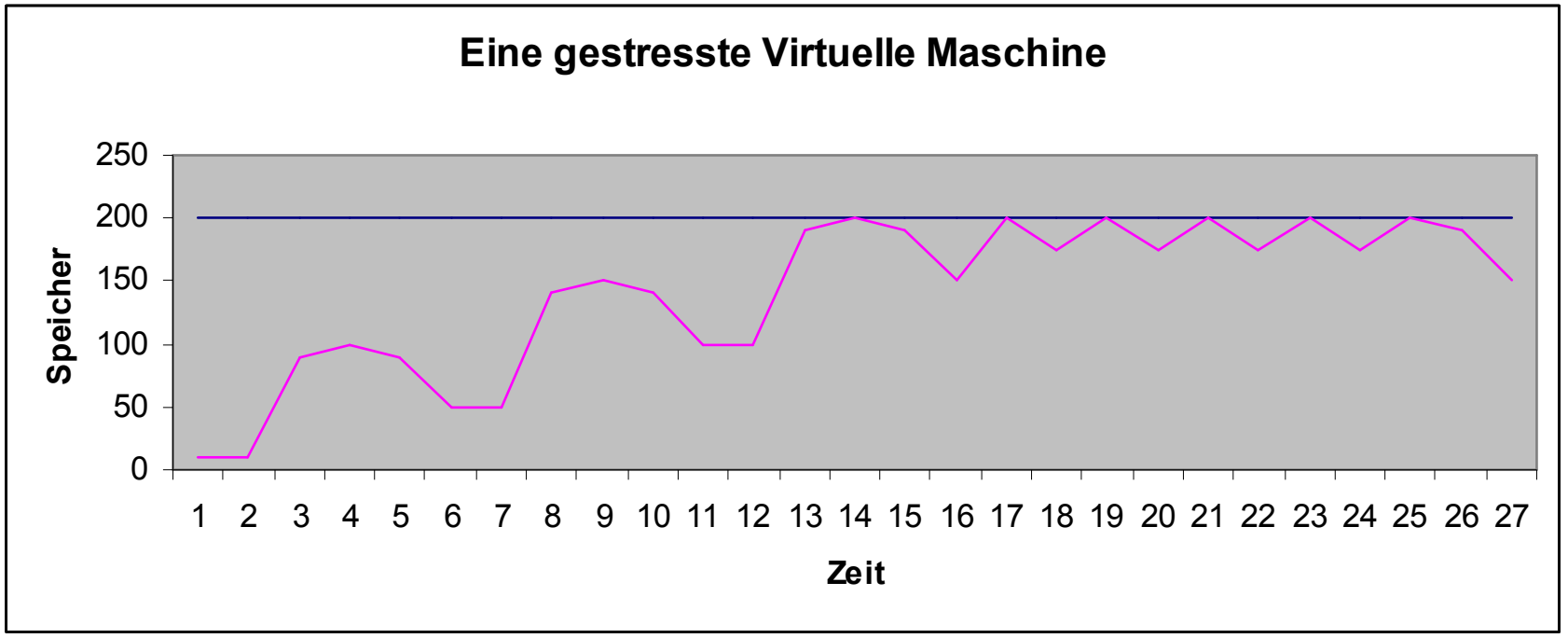


Eine gestresste Virtuelle Maschine

integrata cegos

- Bei den bisher gezeigten Bildern hatte die Virtuelle Maschine kein Problem, den für die Request-Verarbeitung notwendigen Speicher zur Verfügung zu stellen
- Garbage Collections erfolgten jeweils nur am Ende der Request-Verarbeitung bzw. am Ende der Session
- Diese ändert sich jedoch sofort dann, wenn der Speicher die Maximalgrenze erreicht

Eine gestresste Virtuelle Maschine



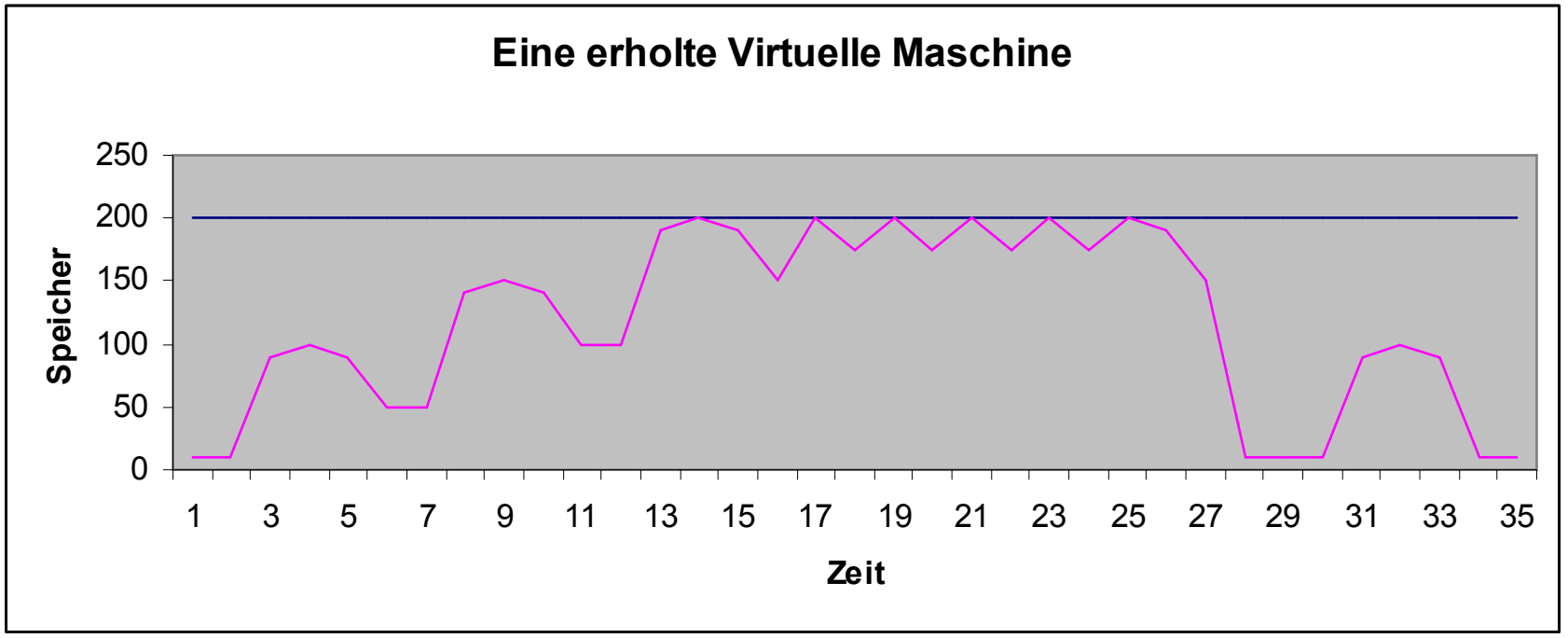
Erläuterung der Stress-Situation

- In diesem Beispiel wird in durch drei aufeinander folgende Requests eine Session von 150 Einheiten aufgebaut
- Die ersten drei Requests erfordern jeweils eine Garbage Collection
- Auf Grund der Speichernot löst der vierte Request nun aber eine ganze Kaskade von Garbage Collections aus, die zwangsläufig zu einer Verlängerung der Verarbeitungsdauer führt
- Aus Sicht des Anwenders dauert also der gleiche Vorgang länger als sonst

Reversible Stress-Situation

- Dieser Stress-Effekt ist reversibel
  - Wird die Session wieder abgebaute ist alles wieder in Ordnung
- Kann während der Stress-Phase für einen Request nicht mehr genügend Speicher angefordert werden, wird diese eine Verarbeitung mit einem OutOfMemoryError abgebrochen
  - Nach der Erholung ist aber alles wieder in Ordnung

Reversible Stress-Situation



integrata cegos

2.12

ALTERUNG DER JAVA VIRTUAL MACHINE

Irreversibler Stress

- Nicht immer kann der Speicher nach Abbau aller Sessions wieder den ursprünglichen Wert erreichen
- Baut die Anwendung beispielsweise einen permanenten Cache auf, den der Programmierer immer weiter befüllt, so wird damit ein Memory Leak erzeugt
  - Die Virtuelle Maschine bleibt dann im gestressten Zustand und kann sich nicht mehr erholen

Erkennen des Memory Leaks

integrata cegos

- Dieser Zustand wird durch Überwachung leicht erkannt
- Die durchschnittliche Antwortzeit des Servers bzw. der Durchsatz sinkt im Laufe der Zeit kontinuierlich ab
- In den Log-Dateien tauchen immer häufiger Request-Abbrüche mit OutOfMemory-Situationen auf
- Eine gealterte Virtuelle Maschine kann sich nur durch Neustart „erholen“

2.13

ANALYSE

Vermeiden von Alterungseffekten

integrata cegos

- Stress- und Alterungseffekte können nicht unbedingt durch eine Server-Konfiguration korrigiert werden
  - Natürlich kann das Auftreten der Probleme durch Einsatz von mehr Speicher zeitlich nach Hinten verschoben werden, aber das Problem tritt irgendwann dann doch auf
- Soll das Problem durch Änderung der Anwendung korrigiert werden ist es sinnvoll, den Entwicklern Informationen darüber zu geben, welche Objekte Speicher blockieren
  - Dies erfolgt durch einen Heap-Dump
  - Ein Administrator kann durch die Anwendung jmap jede Virtuelle Maschine zum Schreiben des Dumps veranlassen und diesen den Entwicklern zukommen lassen

Hinweis zum Speicherdump

- Dump-Dateien sind mindestens so groß wie der Speicher, der gedumped wird
  - Also auf Server-Umgebungen schnell bis zu mehreren Gigabytes
- Das Schreiben des Dumps bedingt zwangsläufig einen sehr ausgeprägten Stop-The-World-Effekt, der durchaus auch mehrere Minuten dauern kann
  - Das Auslösen eines Dumps im Produktionsbetrieb muss deshalb wohl überlegt und vorbereitet werden!

Hinweise zur Performance-Analyse: Hot Spots

integrata cegos

- Die Anwendung startet im interpretierten Modus und wird während ihrer Ausführung analysiert, um die sogenannten „Hot Spots“, also Programmteile, die sehr oft bzw. wiederholt ausgeführt werden, zu finden
- Ist ein Hotspot identifiziert, wird der originale Bytecode des Hotspots durch vom Compiler optimierten nativen Binärcode ersetzt
- Die Art und der Umfang der Optimierung sind dabei abhängig von der Version der Virtuellen Maschine und der verwendeten Variante (Client- oder Servervariante) des Hotspot-Compilers

Beispiele für Hot Spot-Optimierungen

integrata cegos

- Inlining
- Range Check Elimination
- Dead Code Elimination
- Loop Unrolling
- …

integrata cegos

3

MANAGEMENT MIT JMX

3.1

ÜBERBLICK

Einführung

integrata cegos

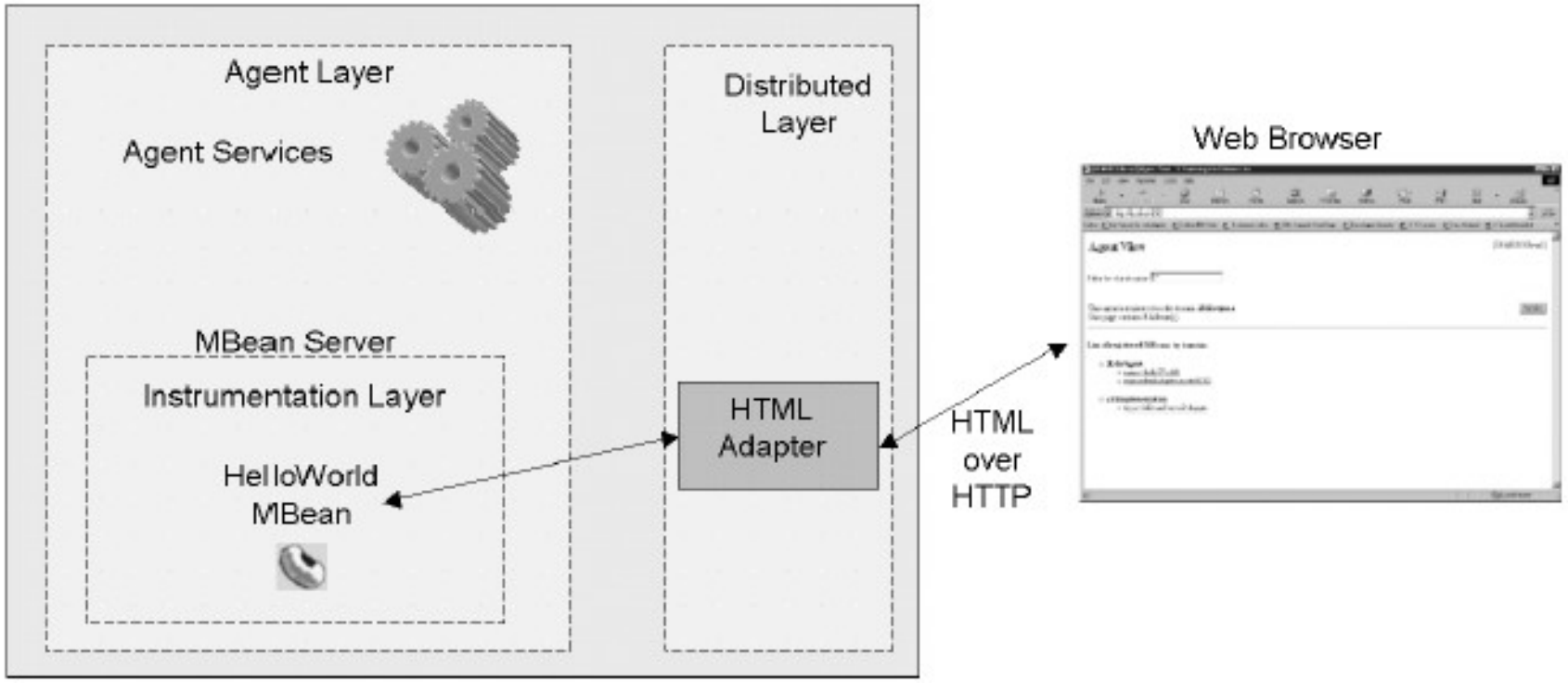
- Java Anwendungen sollen ohne großen Aufwand administriert werden können
  - Insbesondere solle die Lösung durch ein unabhängiges Modul realisiert werden, das vom Rest der Anwendung unabhängig ist
- Um dieses Ziel zu erreichen, definiert die Java Management Extension „JMX“ den JMX Server
- Dieser bietet ist ein Framework bzw. eine Laufzeitumgebung für Managed Beans:
  - Lebenszyklus,
  - Zustandsänderungen,
  - Events und Notification-Mechanismen

JMX-Subsysteme

integrata cegos

- Instrumentation Layer
- Agent Layer
- Distributed Services Layer
  - Innerhalb des Distributed Services Layer werden „normale“ Kommunikationsprotokolle verwendet:
  - http für Web basierte Administrationskonsolen
  - SNMP für vorhandene System
  - Java RMI
  - JINI
  - …

Das JMX-System



3.2

MBEANS

Managed Beans

integrata cegos

- Identifikation über einen eindeutigen Object Name
  - domain:key=value,key2=value2
- Jede Bean hat eine beliebige Menge von Eigenschaften/Attributen
  - Read-only
  - Read/Write
- Jede Bean hat Operationen
  - diese werden direkt im Server ausgeführt
- Der Notification-Mechanismus wird häufig nicht benutzt
  - Dies übernimmt eher ein zentrale Überwachungssoftware wie Nagios

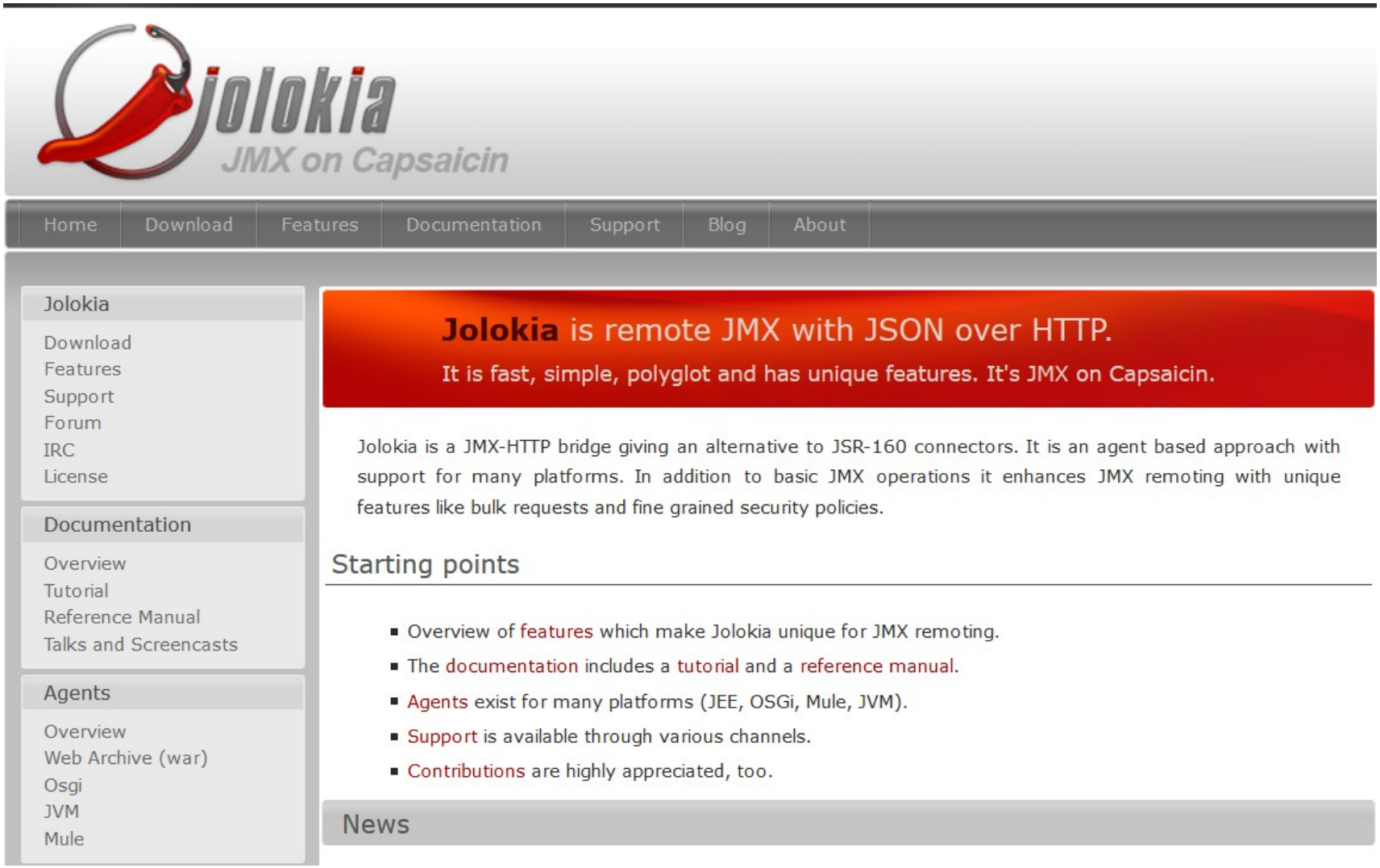
Zugriff auf das JMX-System

integrata cegos

- Spezielle JMX-Clients
  - jconsole als Bestandteil der Java-Installation
- Administrations- und Webkonsolen
  - Mehr oder weniger mächtige Werkzeuge, die die Hersteller von Applikationsservern zur Verfügung stellen
- Übergreifende Produkte
  - Jolokia
    - Eine simple Anwendung, die in jeden Applikationsserver installiert werden kann
    - Zugriff über simple http-Requests
      - cURL!
    - Integration in Überwachungssoftware damit fast trivial
      - HawtIO

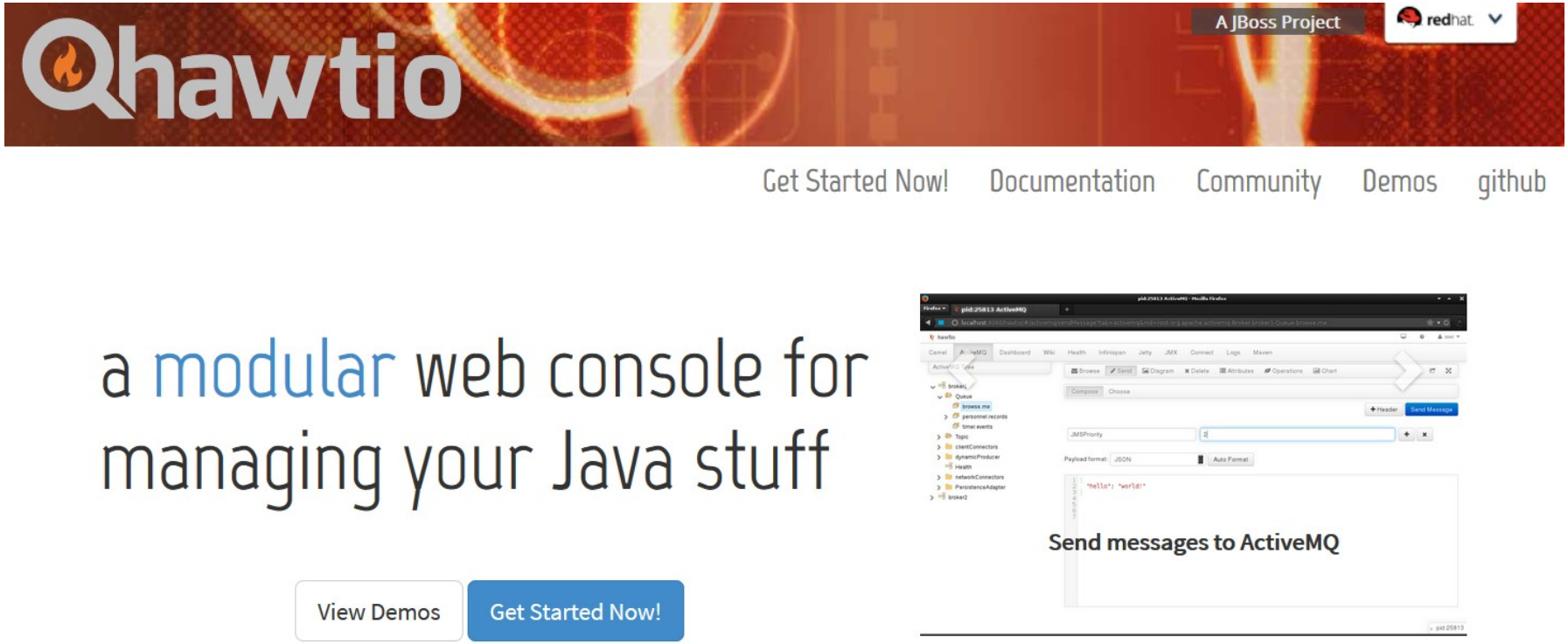
Jolokia

integrata cegos



Hawt.IO

integrata cegos



integrata cegos

4

DER APPLIKATIONSSERVER

integrata cegos

4.1

IT-ARCHITEKTUREN

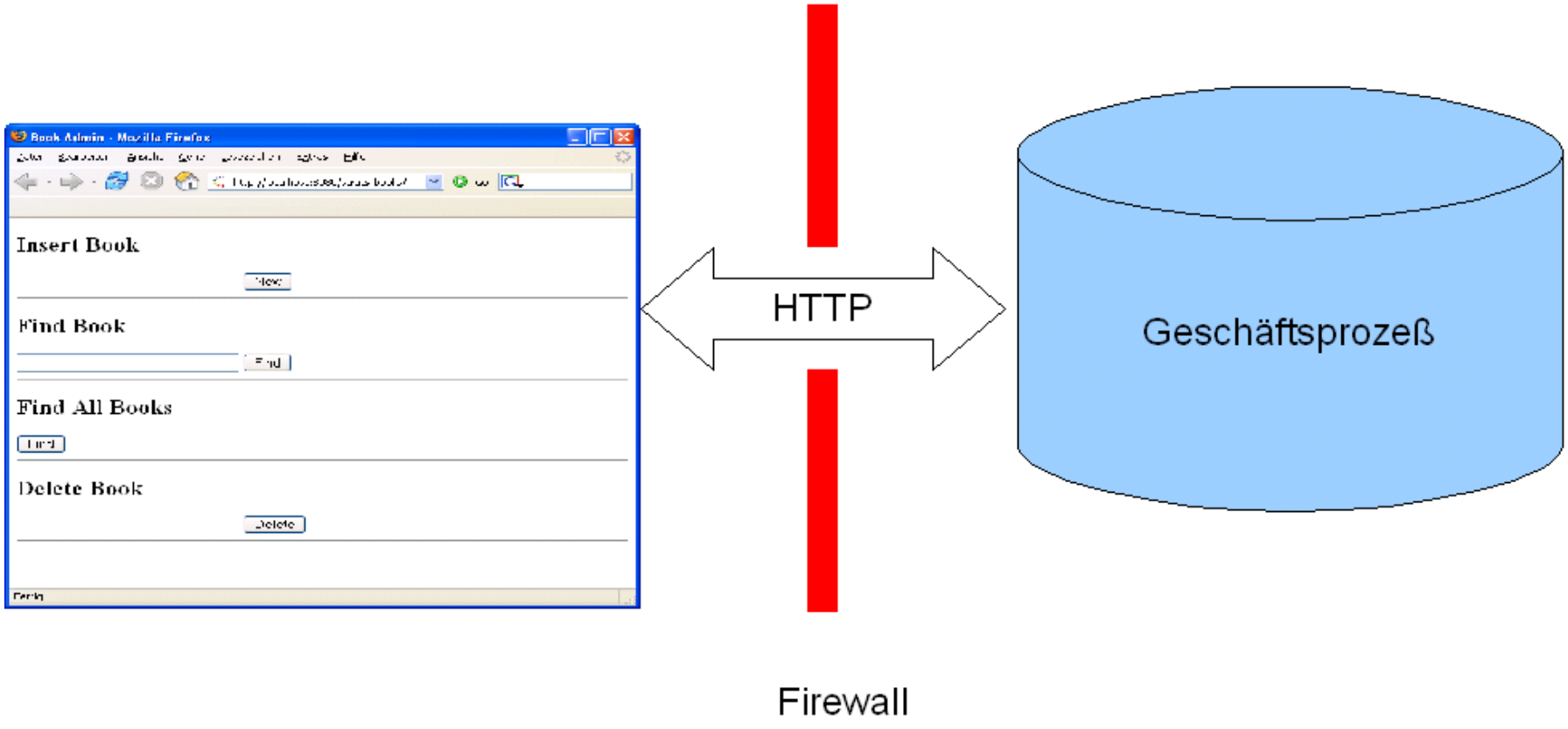
Historie

- Die IT-Landschaft hat sich mit der rasanten Globalisierung der Unternehmen und deren elektronische Kopplung durch standardisierte Protokolle wie das bekannte HTTP innerhalb weniger Jahre massiv verändert
- Früher waren die Unternehmens-internen Prozesse komplett abgeschottet und konnten deshalb relativ problemlos monolithisch und proprietär realisiert werden
  - Klassisches Beispiel hierfür ist eine Großrechner-Anwendung mit interner Datenbank und Terminal-basierten Benutzer-Interaktionen
  - Ein elektronischer Datenaustausch mit den Kunden oder Geschäftspartnern erfolgte entweder gar nicht oder durch spezielle aufgebaute Netzwerke mit jeweils gesondert vereinbarten Protokollen

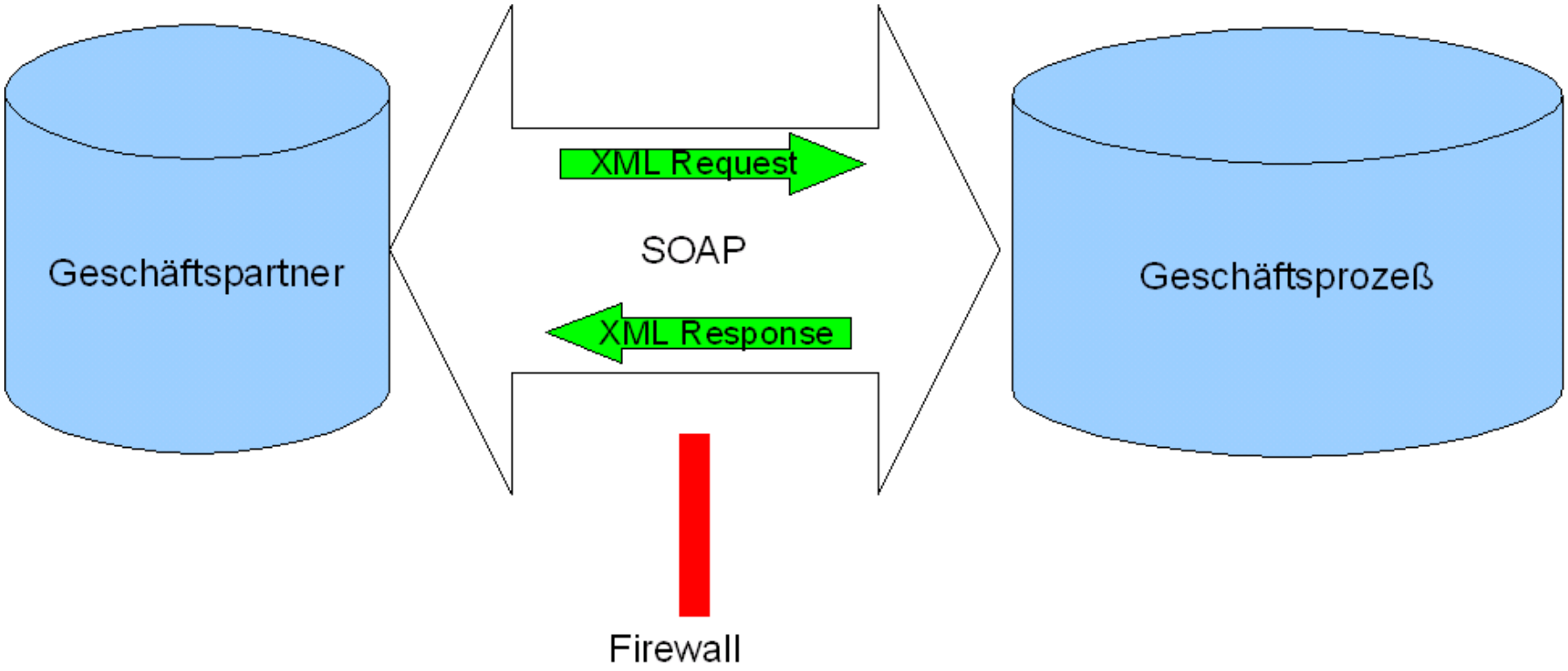
Heute

- Business-to-Consumer (B2C)
  - Bestimmte Geschäftsprozesse werden vom Kunden des Unternehmens direkt aufgerufen
- Business-to-Business (B2B)

Eine typische B2C-Anwendung



Eine typische B2B-Anwendung

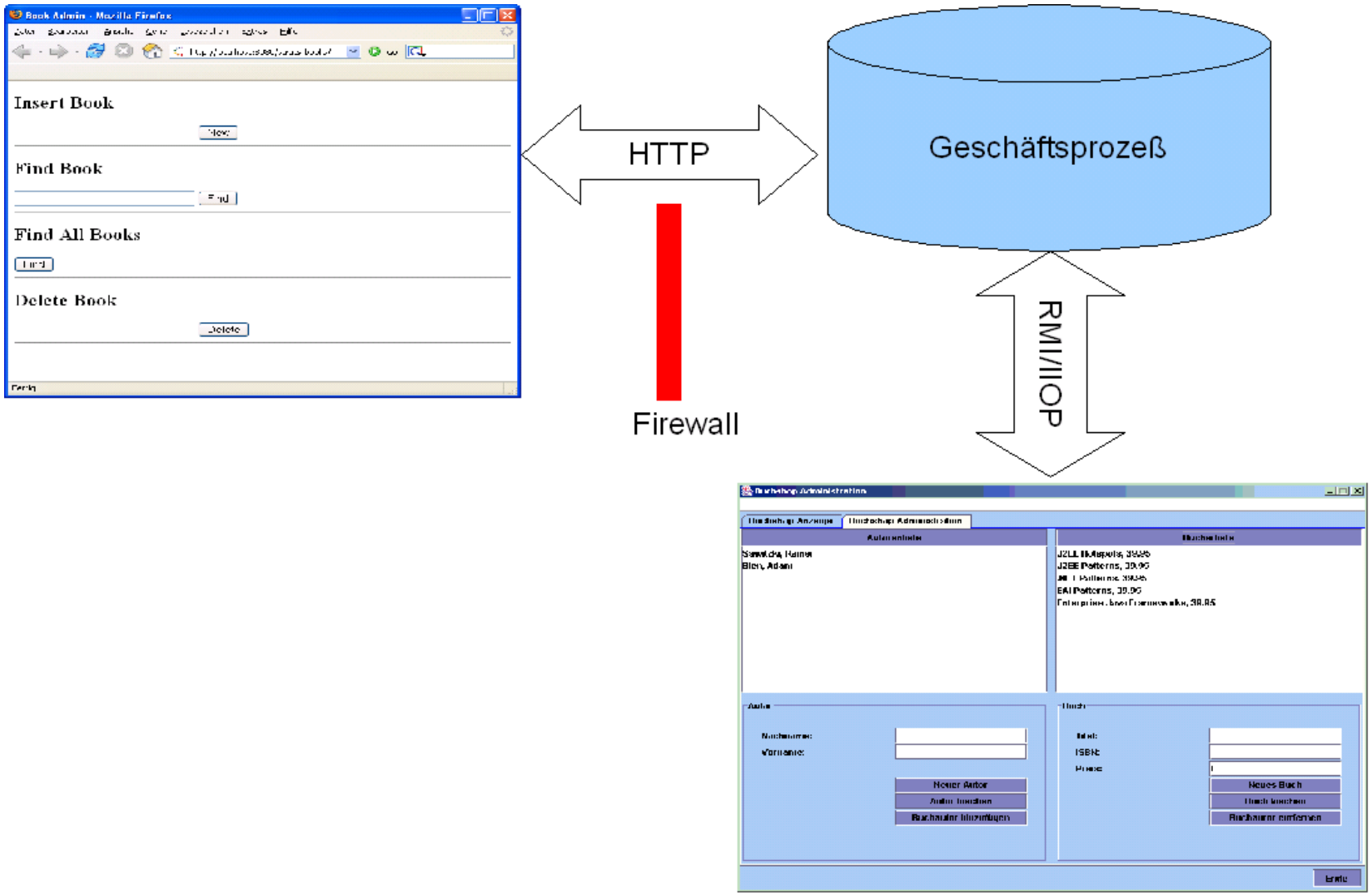


Protokolle

integrata cegos

- Sollen zwei Geschäftsprozesse miteinander kommunizieren und Daten austauschen so sind dafür aktuell viele verschiedene Protokolle möglich, die sich bezüglich Geschwindigkeit, produzierter Netzwerklast und Verlässlichkeit der Verbindung unterscheiden
- Bei Änderungen der Voraussetzungen sollte es einfach möglich sein, ein nun besser geeignetes Protokoll verwenden zu können
  - So kann ein Geschäftsprozess für eine Anwendung, die innerhalb der Firmen-Firewall läuft, problemlos und effizient über eine „Remote Method Invocation“ (RMI) aufgerufen werden
  - Soll der gleiche Prozess aber von Außerhalb aufgerufen werden, wird die Firewall nur für http-Aufrufe durchlässig sein. Wird besonderer Wert auf „Quality of Services“ wie Ausfallsicherheit und Garantie der Übermittlung gelegt so bietet sich eher eine Messaging-Variante an

Protokolle

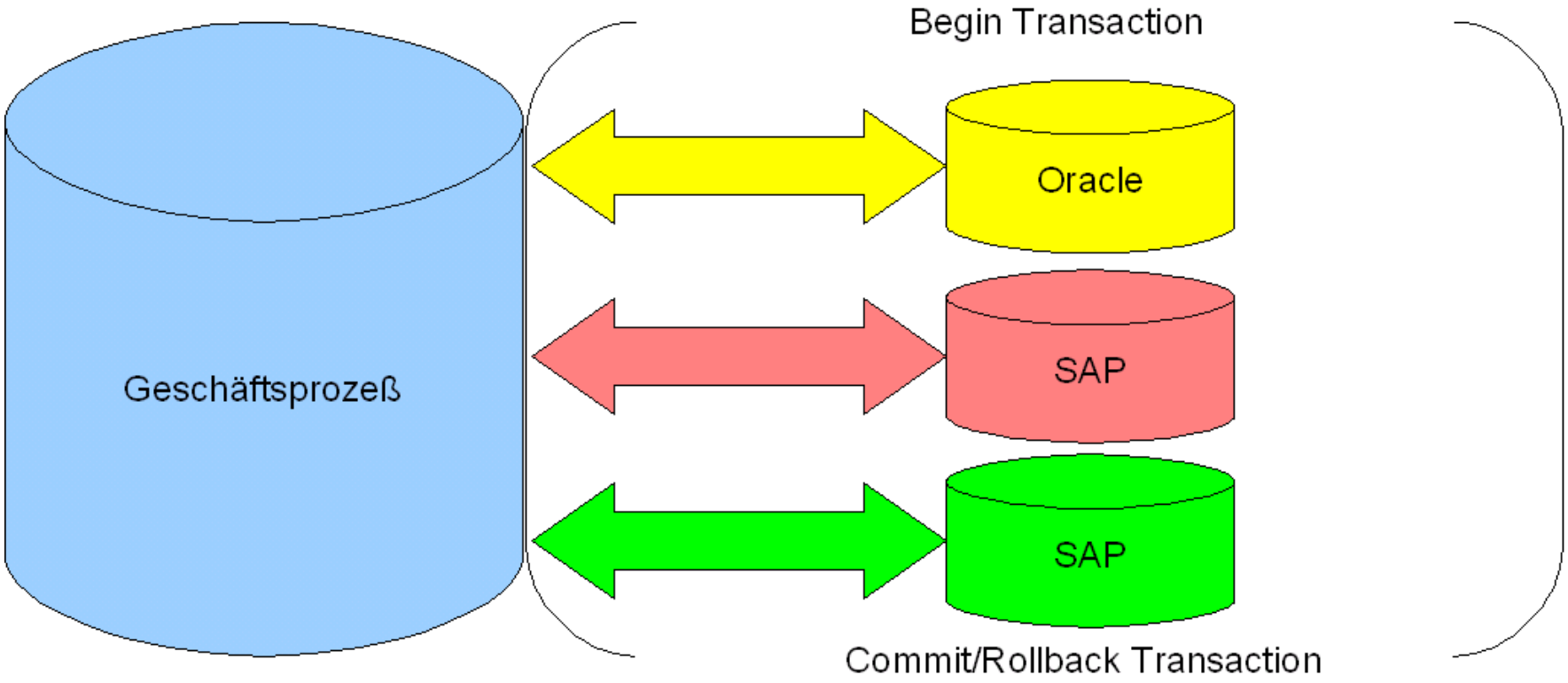

Insert Book
New
Find Book
Find
Find All Books
Find
Delete Book
Delete
HTTP
Geschäftsprozeß
Firewall
RMI/IIOP

Transaktionen

integrata cegos

- Innerhalb eines Unternehmens werden die verschiedensten Betriebssysteme und Software-Systeme parallel betrieben
- Verschiedene Teilprozesse sind oft in verschiedenen Systemen realisiert und müssen von einem zentralen Steuerungssystem angesprochen werden
  - So kann beispielsweise das Bestellen eines Produktes im Hintergrund im Rahmen einer großen Transaktion nacheinander ein Datenbanksystem (z.B. Oracle), SAP und Großrechner-Prozeduren aufrufen
- Für die Umsetzung werden jetzt mehrere Dinge benötigt:
  - Ein Transaktions-Manager (auch geläufig als Transaktionsmonitor), der in der Lage ist, für die verschiedenen System die Transaktion zu kontrollieren
  - Das zentrale Steuerungssystem führt selber wiederum einen Geschäftsprozess aus, der die Aufrufe der anderen Systeme koordiniert

Transaktionen



integrata cegos

4.2

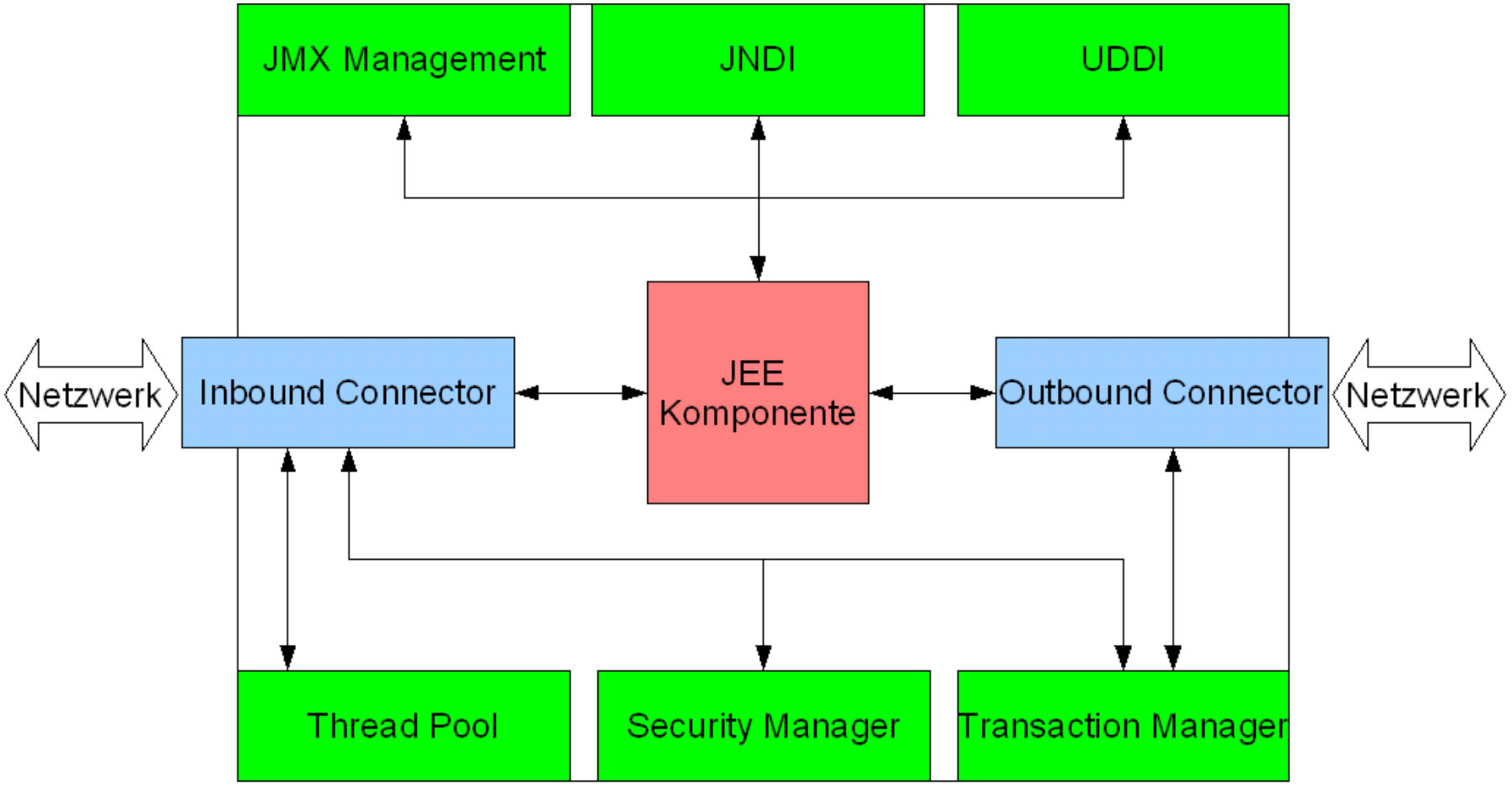
FUNKTIONSWEISE UND KOMPONENTEN DES APPLIKATIONSSERVERS

Dienste

- Jede Aufgabe, die innerhalb des Applikationsservers ausgeführt wird (ein sogenannter „Request“), wird von einem Thread (siehe das folgende Kapitel) exklusiv ausgeführt
- Ein Security Manager übernimmt die Authentifizierung eines Requests und verknüpft die Ausführung mit dem authentifizierten Benutzer
- Der Transaction Manager startet, koordiniert und beendet Ressourcen-übergreifende verteilte Transaktionen
- Alle Komponenten und Dienste werden von einem Management-Dienst (die Java Management Extension, JMX) administriert bzw. überwacht
- Der Applikationsserver dient als Container für Inbound und Outbound Connectors sowie verschiedene Implementierungen von JEE-Komponenten

Der Applikationsserver

integrata cegos



Inbound Connectors

- Inbound Connectors ermöglichen die Kommunikation mit dem Applikationsserver
  - http/https
  - Java Remote Method Invocation (RMI) bzw. die interoperable Variante RMI/IIOP
  - Empfangen von Messages mit dem Java Messaging Service (JMS)
  - SOAP
- Für Protokolle, die nicht im Standard enthalten sind, können spezielle Implementierungen für Inboud Connectors als Plug-In in den Applikationsserver installiert werden
  - Dafür existiert auch ein breiter Markt von Herstellern und Produkten

Aufgaben des Inbound Connectors

integrata cegos

- Die Inbound Connectors enthalten selber keine Geschäftsprozesse
- Laut Konzeption wandeln sie eine einkommende Netzwerkverbindung um in einen Request der, wie oben bereits angesprochen, von einem Thread aus dem Thread Pool ausgeführt wird
- Geschäftslogik wird in einer JEE-Komponente definiert
- Ein Inbound Connector kann im Rahmen einer Verteilten Transaktion aufgerufen werden

JEE Komponenten

- JEE Komponenten bestehen aus zwei Komponenten:
  - Einem Container, der vom Applikationsserver zur Verfügung gestellt wird
  - und der eigentlichen Komponente, die als Bestandteil eines Anwendungsprogramms aufzufassen ist

JEE Komponenten

integrata cegos

- Servlets werden vom http Inbound Connector aufgerufen
  - Die Entscheidung, welches Servlet (und damit welche Anwendungslogik) aufgerufen werden soll erfolgt über die URL des http-Aufrufs
  - Ein spezieller Typ von Servlet kann auch über SOAP aufgerufen werden
  - In modernen Anwendungen existiert wahrscheinlich nur ein einziges zentrales Servlet
- Enterprise JavaBeans dienen primär zur Transaktionssteuerung
  - Dem Anwendungsprogrammierer werden verschiedene Sub-Typen angeboten, die sich bezüglich Lebenszyklus und Aufruf-Protokoll unterscheiden
  - Es gibt „Stateless“ und „Stateful SessionBeans“, „Message Driven Beans“ sowie „Singletons“
  - Enterprise JavaBeans sind in modernen Anwendungen häufig nicht mehr anzutreffen

Outbund Connectors

- Outbound Connectors werden von JEE-Komponenten lokal benutzt
- Ihre Aufgabe ist es, einen Geschäftsprozess innerhalb eines Backend-Systems aufzurufen
  - Dazu hält der Outbound Connector einen Connection Pool
- Die Outbound Connectors werden auch vom Transaktionsmanager (für die Koordinierung der Verteilten Transaktionen) und vom Security Manager (Propagierung des angemeldeten Benutzers) benutzt

Beispiele für Outbound Connectors

integrata cegos

- Die JEE-Spezifikation verlangt die folgenden Outbound Connectors:
- Die Data Sources verbinden zu SQL-Datenbanksystemen
- Versenden von JMS Messages
- Anbinden an ein Email-System
- Alle weiteren Systeme sind wiederum durch Plug-Ins installierbar, für die ein breiter Markt existiert

4.3

DATASOURCES

DataSources

integrata cegos

- Die wahrscheinlich am häufigsten benutzte Ausprägung eines Outbound Connectors ist die DataSource
- Damit wird eine Datenbank angebunden
- Die Konfiguration erfolgt meistens über eine Datenbank-URL
- Diese setzt sich aus folgenden Informationen zusammen:
  - Protokoll, definiert das Protokoll, mit dem auf die Datenbank zugegriffen wird
  - Server, Host-Adresse
  - Port, Portnummer
  - Datei-/Datenbankname, Datei, in der sich die Datenbank befindet oder symbolischer Name der Datenbank
  - Parameter

Protokoll

integrata cegos

- Das Protokoll besteht aus zwei Teilen,
  - dem Hauptprotokoll, das immer jdbc: heißt und
  - einem Subprotokoll, das sich nach der verwendeten Datenbank richtet
  - Es trägt daher häufig den Namen der Datenbank, des Datenbank-Herstellers oder des Treibers
  - Beispiele sind odbc:, cloudscape:rmi:, borland:dsremote: etc.
- Das Protokoll sieht also immer so aus:
  - jdbc:subprotokoll:

Server und Port

- An das Protokoll können sich eine Server- und eine Portbezeichnung anschließen
  - Sie ist nicht notwendig, wenn sich die Datenbank auf demselben Rechner befindet, wie das Java-Programm oder, wenn sie mit einem Alias über einen Dienst angesprochen wird
- Der Server wird durch eine IP-Adresse spezifiziert, entweder in der Punktnotation, z. B. 67.8.78.24 oder durch einen Domänennamen, z. B. www.integrata.de
- Der Port folgt auf die IP-Adresse durch einen ':' getrennt
- Server und Port haben folgende vollständige Notation
  - //host:port

Datei- oder Datenbankname

- Unbedingt erforderlich ist natürlich der Datenbankname
- Wie dieser zu spezifizieren ist, hängt von dem jeweiligen Hersteller ab
  - Es gibt daher keine allgemein gültige Regel für den Datenbanknamen

Parameter

- Nach dem Datenbanknamen können auch noch Parameter spezifiziert werden
  - Auch hier gibt es keine festen Regeln, wie diese Parameter auszusehen haben
  - Die Hersteller sind frei in der Auswahl und der Formatierung ihrer Parameter
  - Üblicherweise werden die Parameter durch ';' getrennt
- Beispielsweise gestattet die Derby-Datenbank folgenden Parameter
  - ;create=true
- wenn eine Datenbank, sofern sie nicht existiert, angelegt werden soll, bzw.
  - ;create=false
- wenn eine vorhandene Datenbank verwendet werden soll

Vollständige URL

- Eine vollständige Datenbank-URL setzt sich also folgendermaßen zusammen
- jdbc:subprotokoll:[//host]:[port]/dbName[;parameter]

Benutzer und Kennwort

- Wenn auf einer Datenbank Benutzer mit Kennwörtern eingerichtet sind, so können diese beim Öffnen der Verbindung spezifiziert werden
- Sowohl der Benutzer, als auch das Kennwort werden als String übergeben

integrata cegos

5

WERKZEUGE IM JAVA-UMFELD

5.1

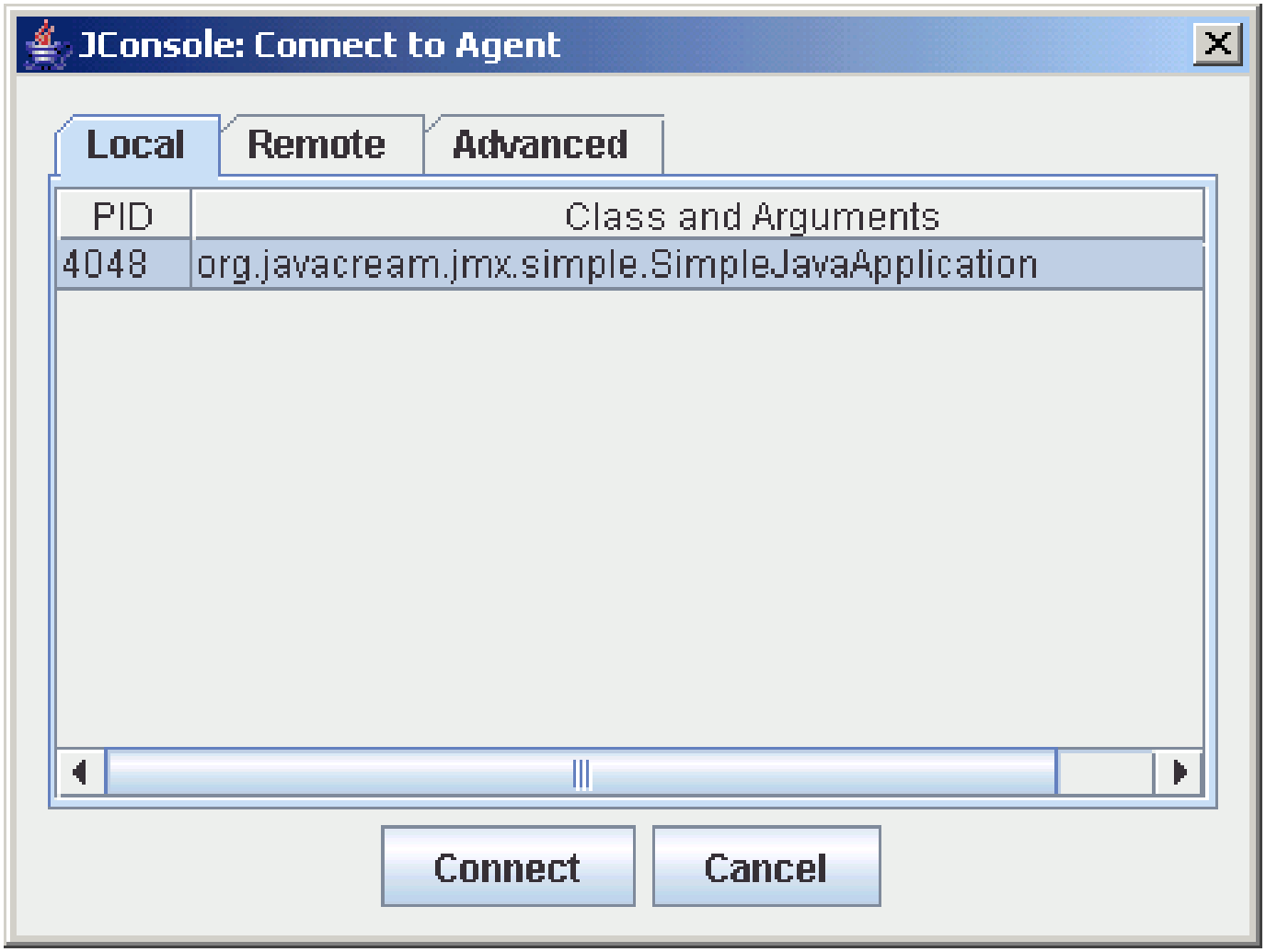
DIE JCONSOLE

Die jconsole

- Das JRE beinhaltet auch eine visuelle Oberfläche um die JMX-Objekte der laufenden Java-Prozesse darstellen zu können
- Beim Aufruf dieses Programms versucht es, alle lokal laufenden überwachbaren Java-Prozesse zu finden und stellt die in einem Auswahlfenster dar

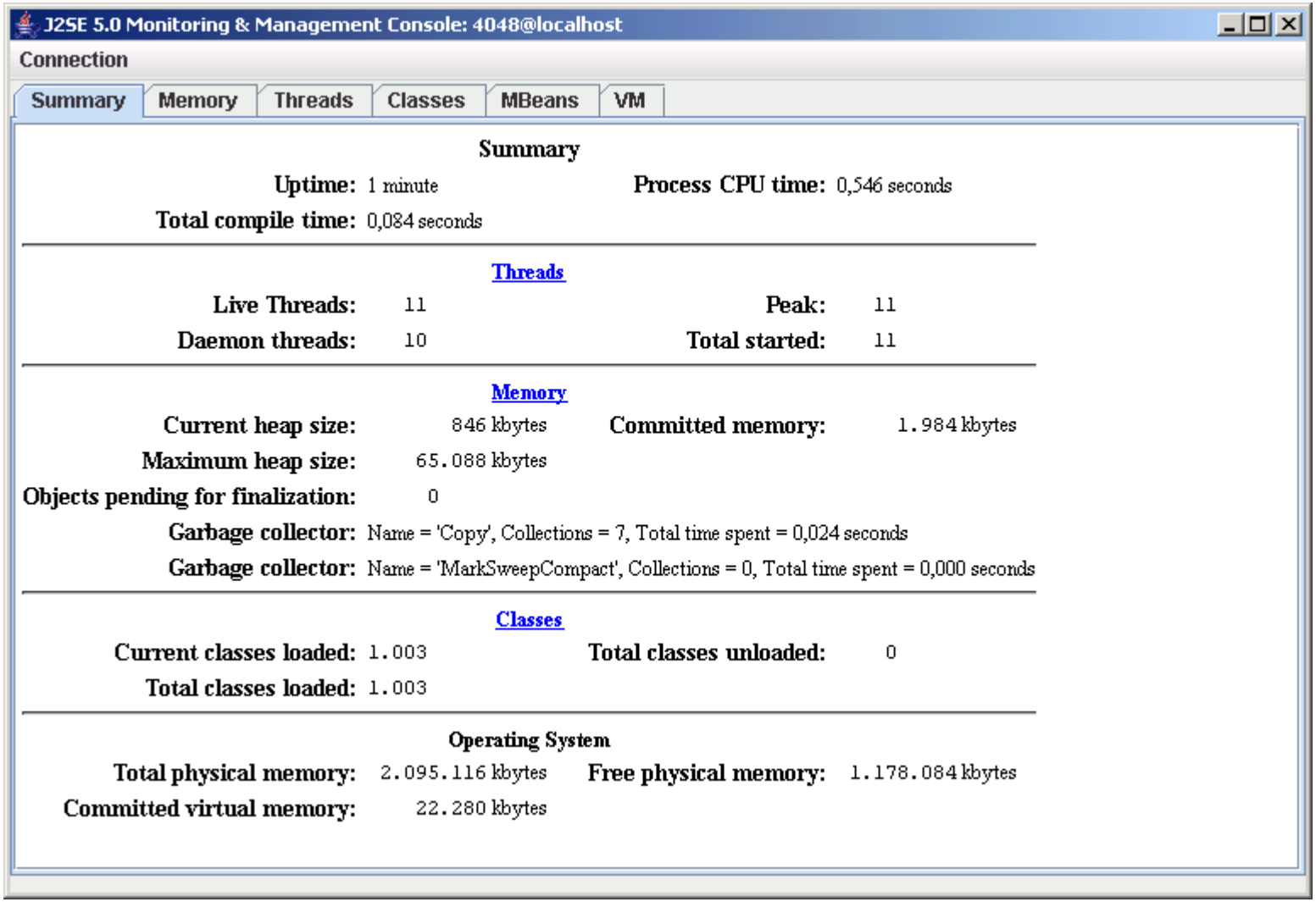
Connect-Dialog der jconsole

integrata cegos



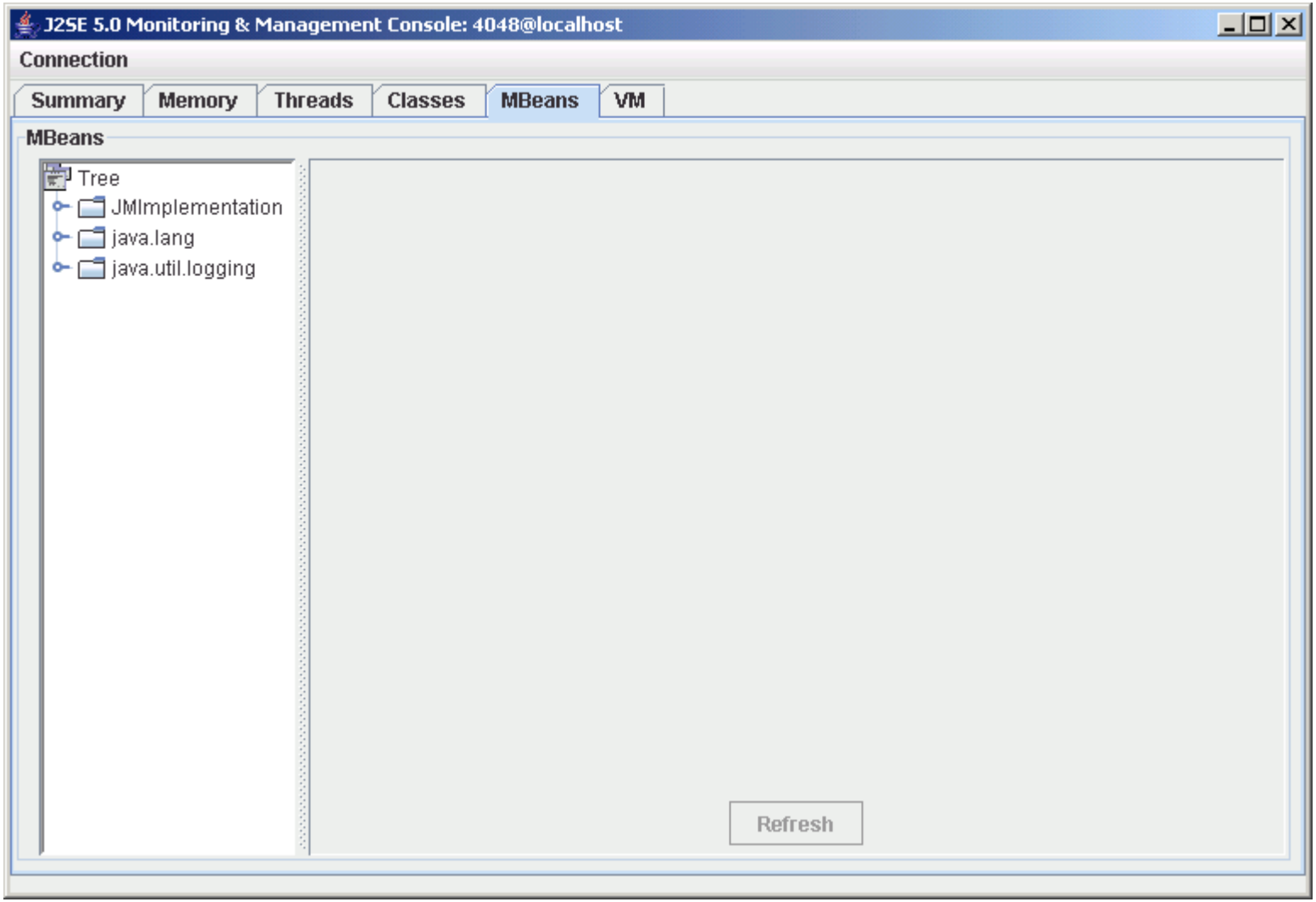
Oberfläche

integrata cegos



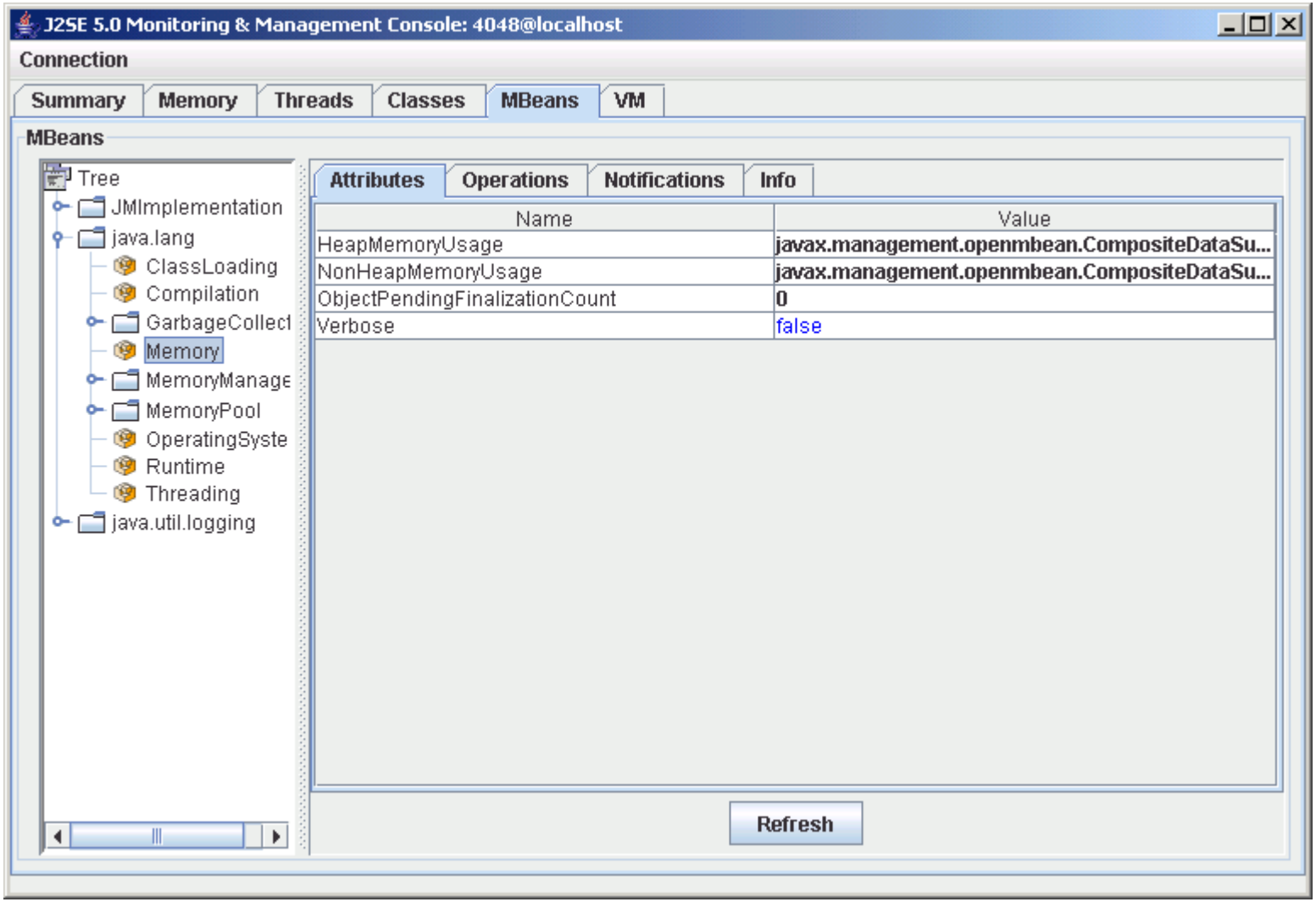
MBeans-Reiter

integrata cegos



Attribute und Operationen

integrata cegos



integrata cegos

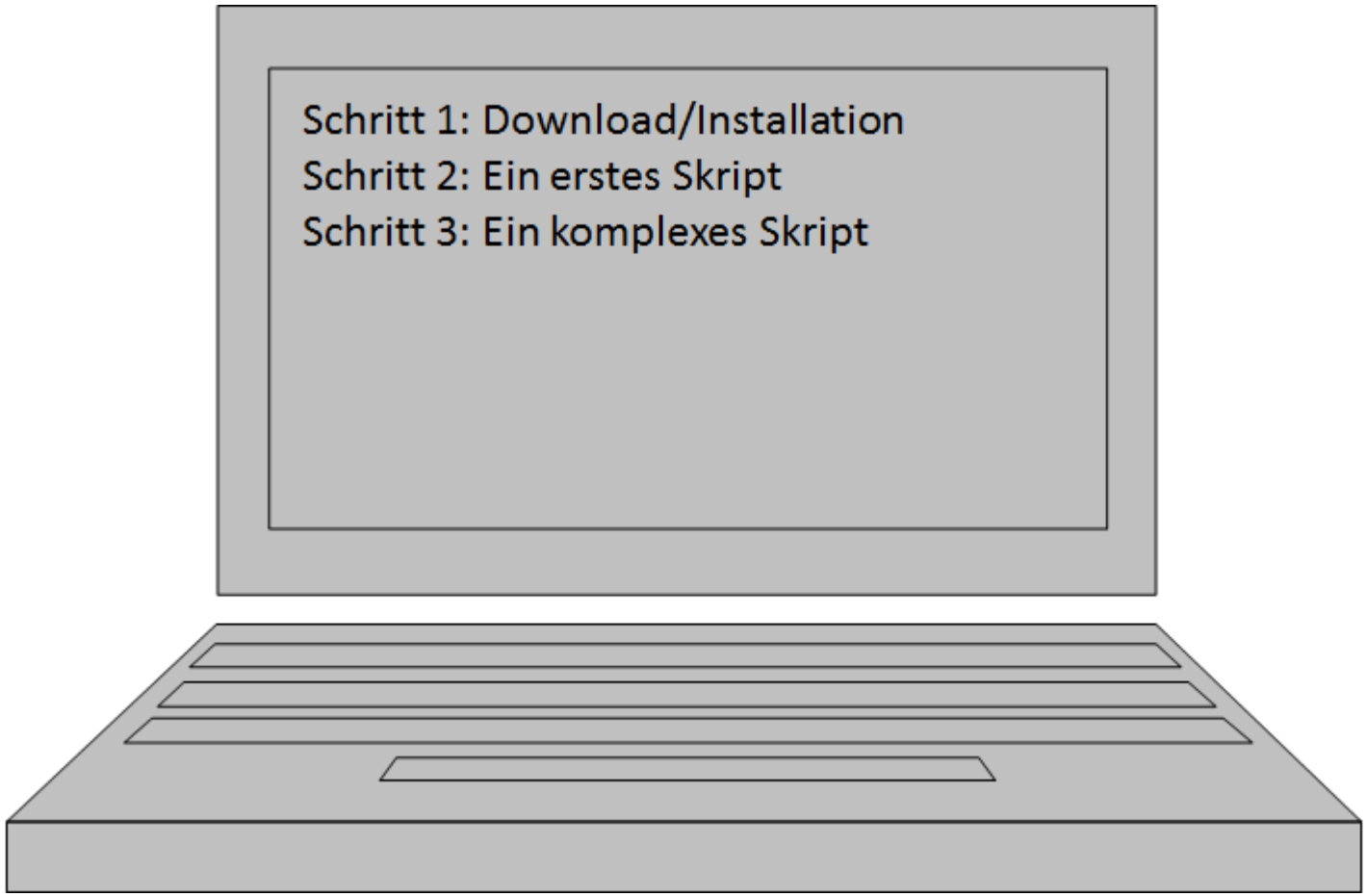
5.2

APACHE ANT

Allgemeines

- Ant ist eine Skript-Sprache, eine ausführliche Online-Dokumentation, ist unter: http://ant.apache.org/manual/index.html zu finden
- Der Skript-Prozess wird bei Ant über XML-Dateien gesteuert, in denen Projekte, Abhängigkeiten und Arbeitsschritte in Form von Tasks definiert werden
  - Die eigentliche Ausführung erfolgt dann in einer Java Virtual Machine, es muss ein JDK installiert sein

In drei Schritten zum Buildfile



integrata cegos

5.3

APACHE JMETER

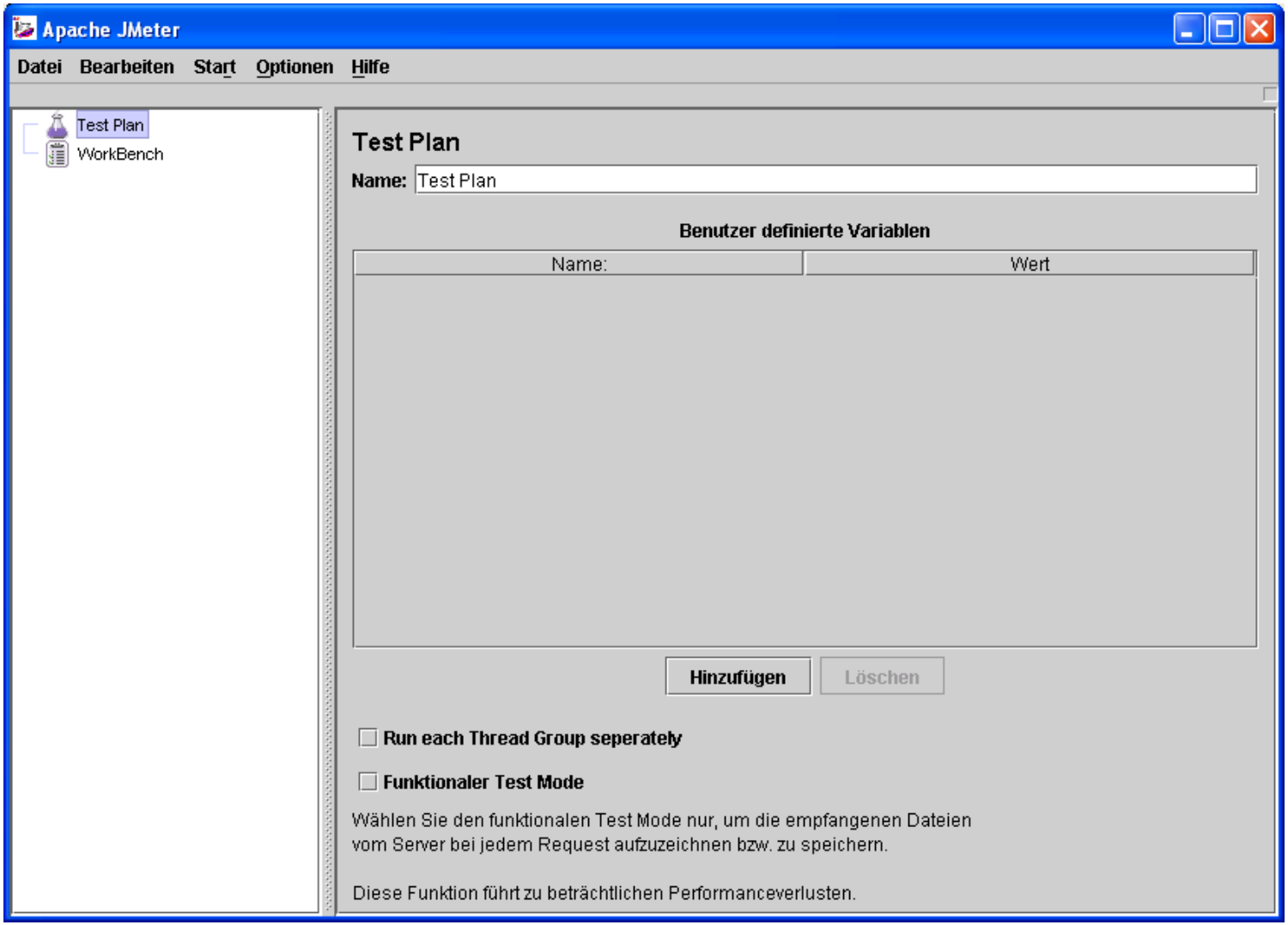
Übersicht

integrata cegos

- Dieses Apache Framework dient zur Performance- und Last-Analyse von Java Anwendungen
- Der Focus des Projekts lag auf Web Anwendungen, umfasst durch ein Plug-In-Konzept jedoch mittlerweile auch den Zugriff auf Datenbanken und andere TCP/IP-basierte Protokolle
- Im Gegensatz zu HttpUnit dienen die Tests mit JMeter jedoch vorwiegend zur Erzeugung von Last auf den Server
- JMeter wird benutzt als eigenständige Gui-Anwendung, kann aber auch in Ant integriert werden

Oberfläche

integrata cegos



Der Testplan

- Um JMeter zu benutzen, muss ein TestPlan ausgeführt werden
  - Dieser besteht wiederum aus einem oder mehreren TestGroups
  - Jede Gruppe simuliert eine Menge von Benutzern
- Fertige Testpläne werden im oberen Bereich der Oberfläche angezeigt, Pläne im Bereich in der Workbench

Details zum Testplan

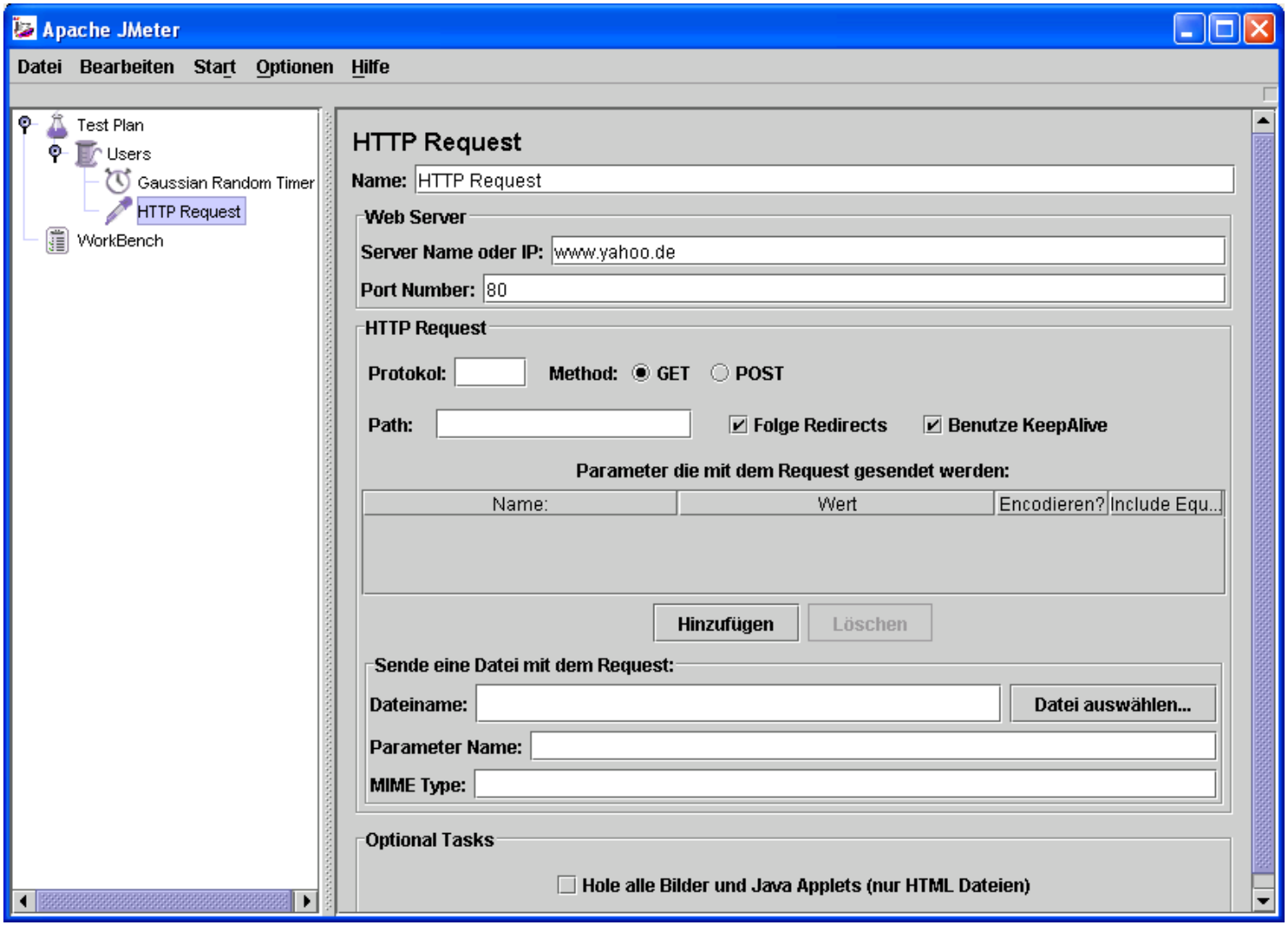
- Ein Testplan besteht wiederum aus einzelnen Subelementen:
  - Ein Timer definiert die Zeit, die ein Thread innerhalb der ThreadGroup bis zum nächsten Aufruf wartet
    - Ist kein Timer gesetzt, so läuft der Thread permanent
  - Controller stehen in zwei Ausprägungen zur Verfügung: Samplers und Logik
    - Der erstere definiert einen bestimmten Test und stellt Daten zur Verfügung, der zweite definiert den Kontrollfluss zwischen den enthaltenen Subcontrollern
- Ein Listener empfängt die Daten der Controller und stellt sie entweder grafisch dar oder speichert sie ab

Controller

- Als Sampler stehen zur Verfügung:
  - FTP Request
  - HTTP Request
  - JDBC Request
  - Java object request
  - LDAP Request
  - SOAP/XML-RPC Request
  - WebService (SOAP) Request
- Sampler müssen ebenso wie die ThreadGroup noch konfiguriert werden

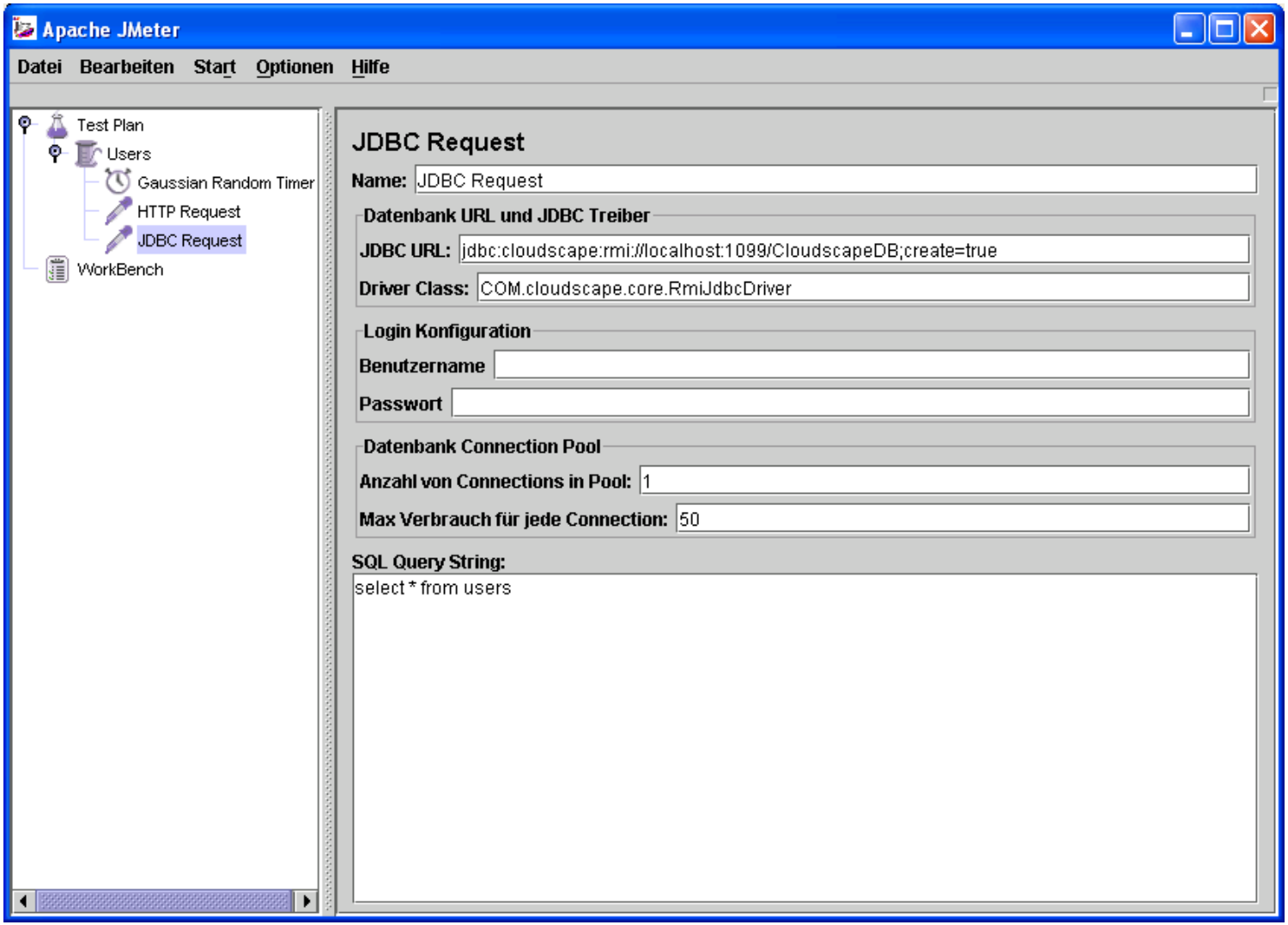
Beispiel: Http Sampler

integrata cegos



Beispiel: JDBC Sampler

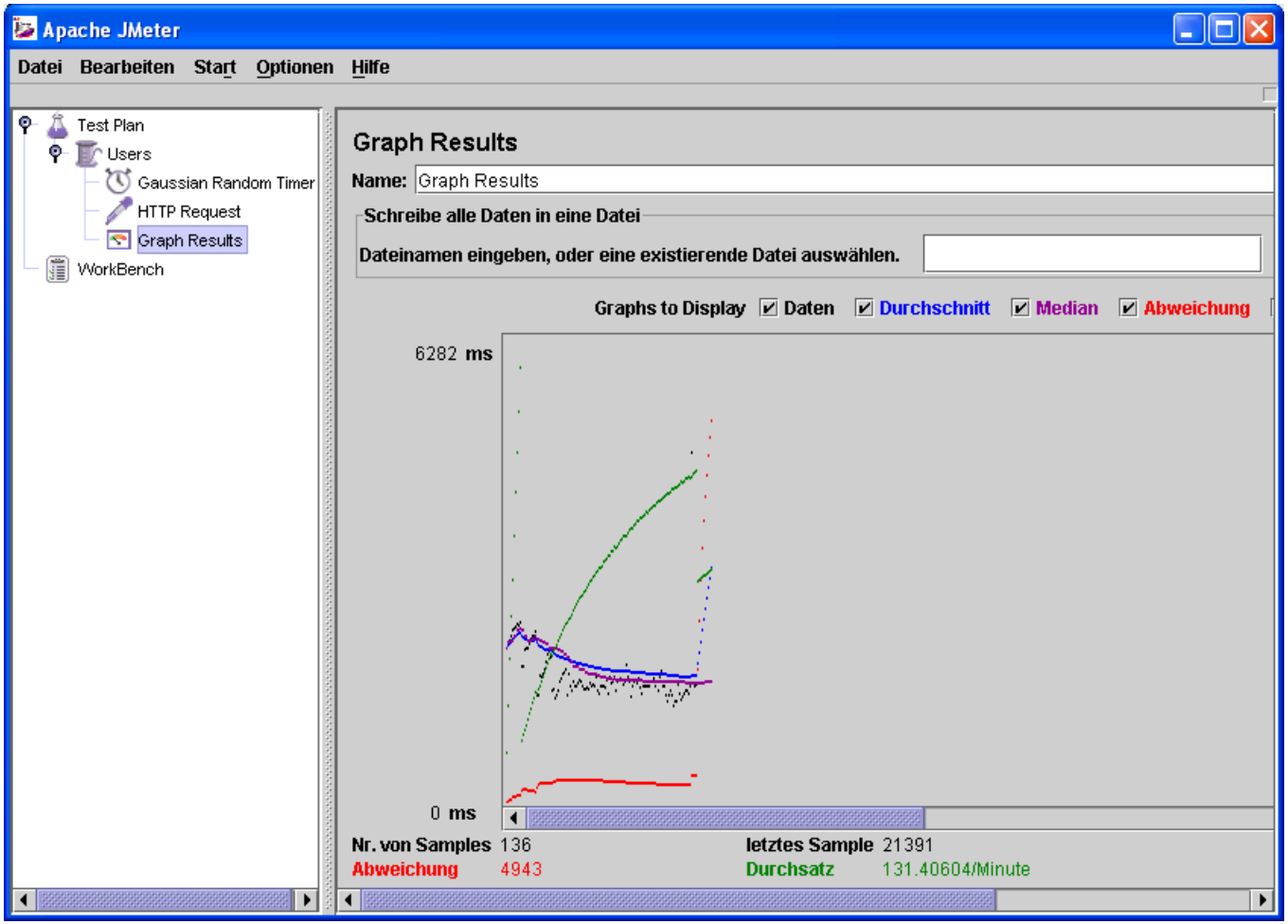
integrata cegos



Logische Controller

- Interleave Controller (Aus Kontroller)
  - Die Subcontroller des Interleave Controllers werden von diesem alternierend für jeden Schleifendurchlauf der ThreadGroup aufgerufen
- Loop Controller (Wiederholungskontroller)
  - Der Loop Controller iteriert über die enthaltenen Subcontroller unabhängig von der Anzahl der Durchläufe der ThreadGroup
- Once Only Controller (Nur einmal Kontroller)
  - Die Subcontroller werden während der Iterationen über die ThreadGroup nur ein einziges mal ausgeführt
- Simple Controller (Einfacher Kontroller)
  - Diese dienen nur Erhöhung der Übersichtlichkeit
- Random Controller (Zufalls Kontroller)
  - Ähnlich wie der Interleave Controller werden die enthaltenen Subcontroller in zufälliger Auswahl aufgerufen
- Recording Controller (Recording Controller)
  - Ein Platzhalter der anzeigt, wo während der Verwendung eines http Proxies die Daten gesammelt werden sollen
- Module Controller (Module Kontroller)
  - Der Module Controller dient als Platzhalter für Subcontrollern
- Throughput Controller (Throughput Controller)
  - Hier kann angegeben werden, wie oft die Subcontroller während der Iteration über die ThreadGroup ausgeführt werden

Listener


Apache JMeter
Datei Bearbeiten Start Optionen Hilfe
Test Plan
Users
Gaussian Random Timer
HTTP Request
Graph Results
WorkBench
Graph Results
Name: Graph Results
Schreibe alle Daten in eine Datei
Dateinamen eingeben, oder eine existierende Datei auswählen.
Graphs to Display Daten Durchschnitt Median Abweichung
6282 ms
0 ms
Nr. von Samples 136
letztes Sample 21391
Abweichung 4943
Durchsatz 131.40604/Minute

Übersicht Listener

integrata cegos

- Mailer Visualizer
  - Wird eine einstellbare Menge von Fehlern registriert, wird eine Mail gesendet
- Graph Full Result; Graph Results
  - Darstellung der Ergebnisse in grafischer Form
- Spline Visualizer
  - Hier werden die Ergebnisse des Tests stets auf 100 Durchläufe normiert angezeigt
- Assertion Results
  - Eine einfache Auflistung aller Tests, gegebenenfalls mit aufgetretenen Fehlern
- View Result Tree
  - Eine Auflistung aller Tests in Tree-Form, gegebenenfalls mit aufgetretenen Fehlern
- Aggregate Report
  - Tabelle mit den summierten Ergebnissen der Tests
- View Results in Table
  - Tabelle mit den Ergebnissen der Tests pro Durchlauf
- Simple DataWriter
  - Ausgabe der Ergebnisse in eine Datei

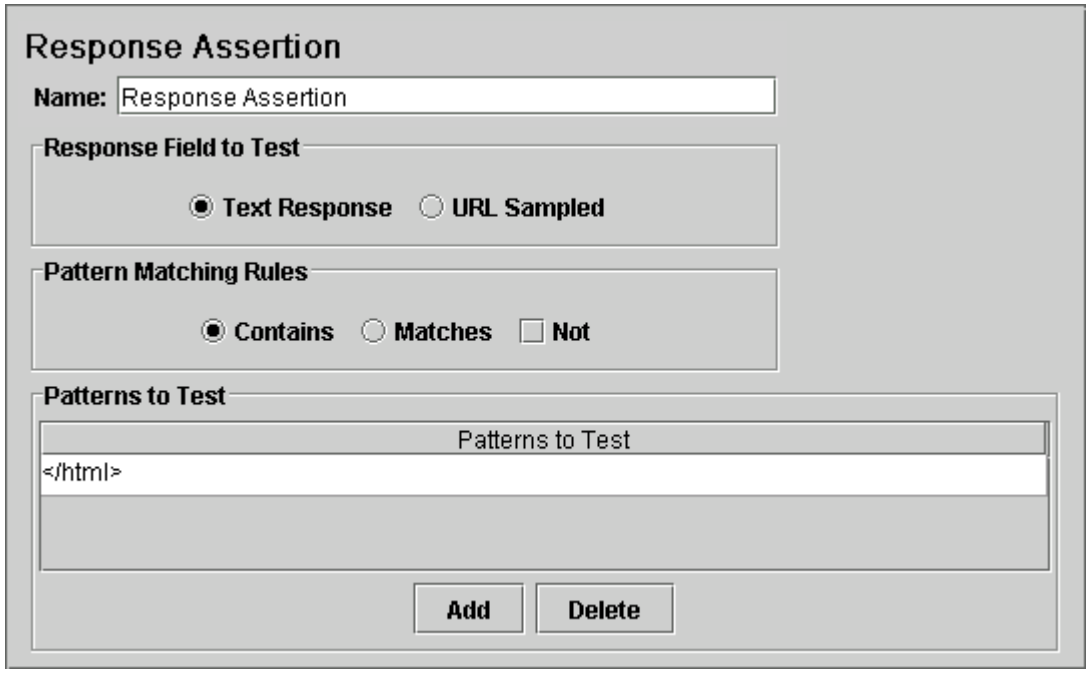
Timers

- Constant Timer
- Gaussian Random Timer
- Uniform Random Timer
- Constant Throughput Timer
  - dieser versucht, die Gesamtpausenzeit einer Vorgabe gemäß anzupassen

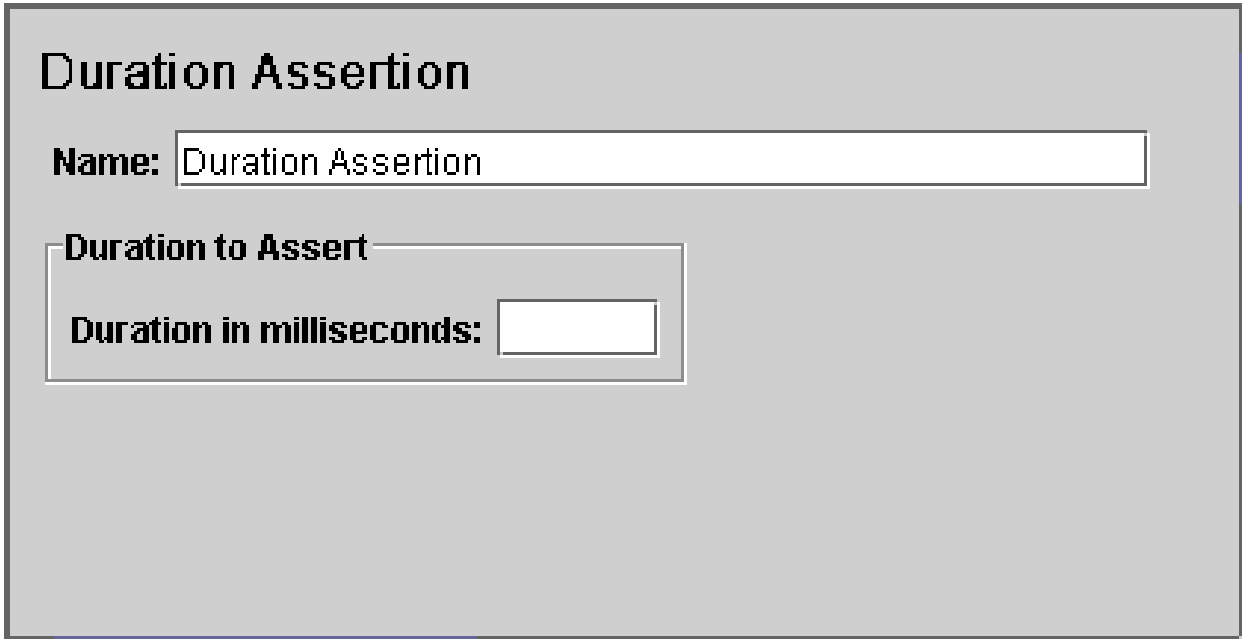
Assertions

- Ein Controller kann Assertions beinhalten, die als Tests aufgefasst werden können
- Assertions können so konfiguriert werden, dass der Lasttest abgebrochen wird
- Vorsicht:
  - Mit Assertions können auch fachliche Tests formuliert werden
  - Dies ist jedoch nicht der Primärfokus von JMeter

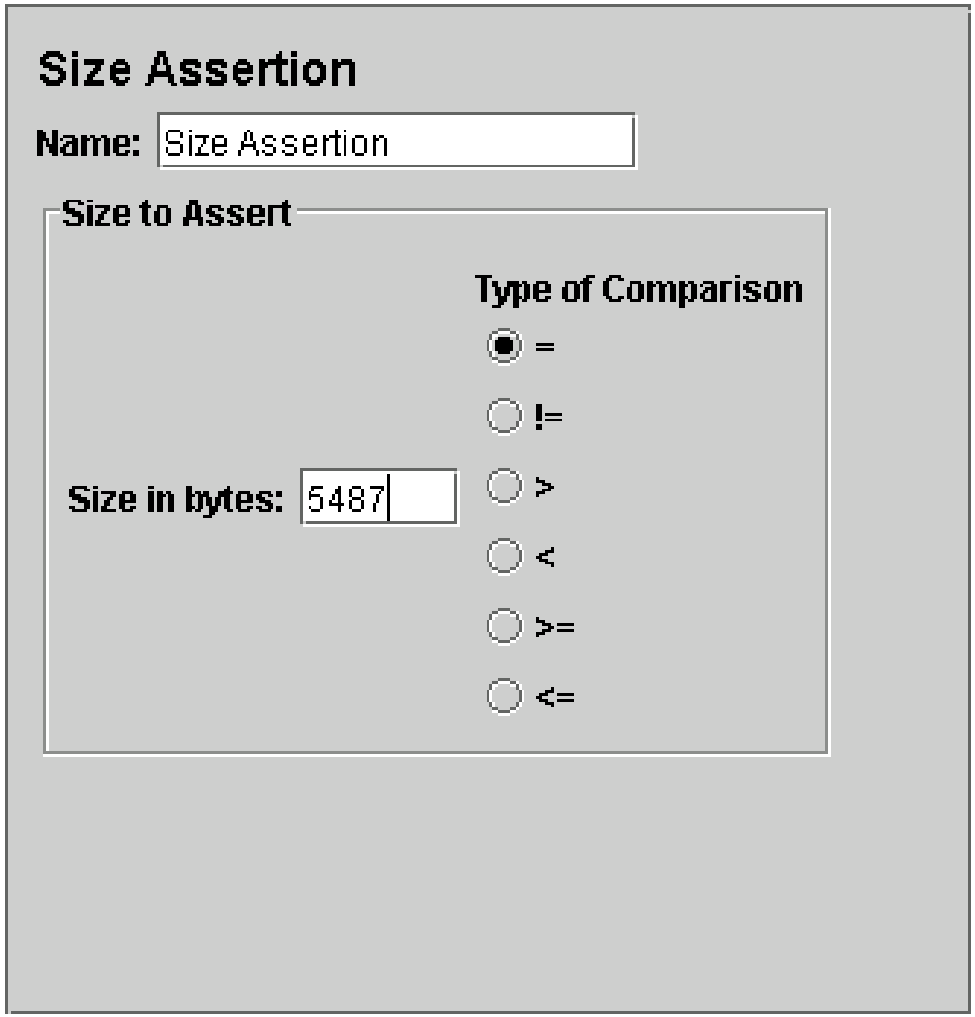
Response Assertion



Duration Assertion



Size Assertion



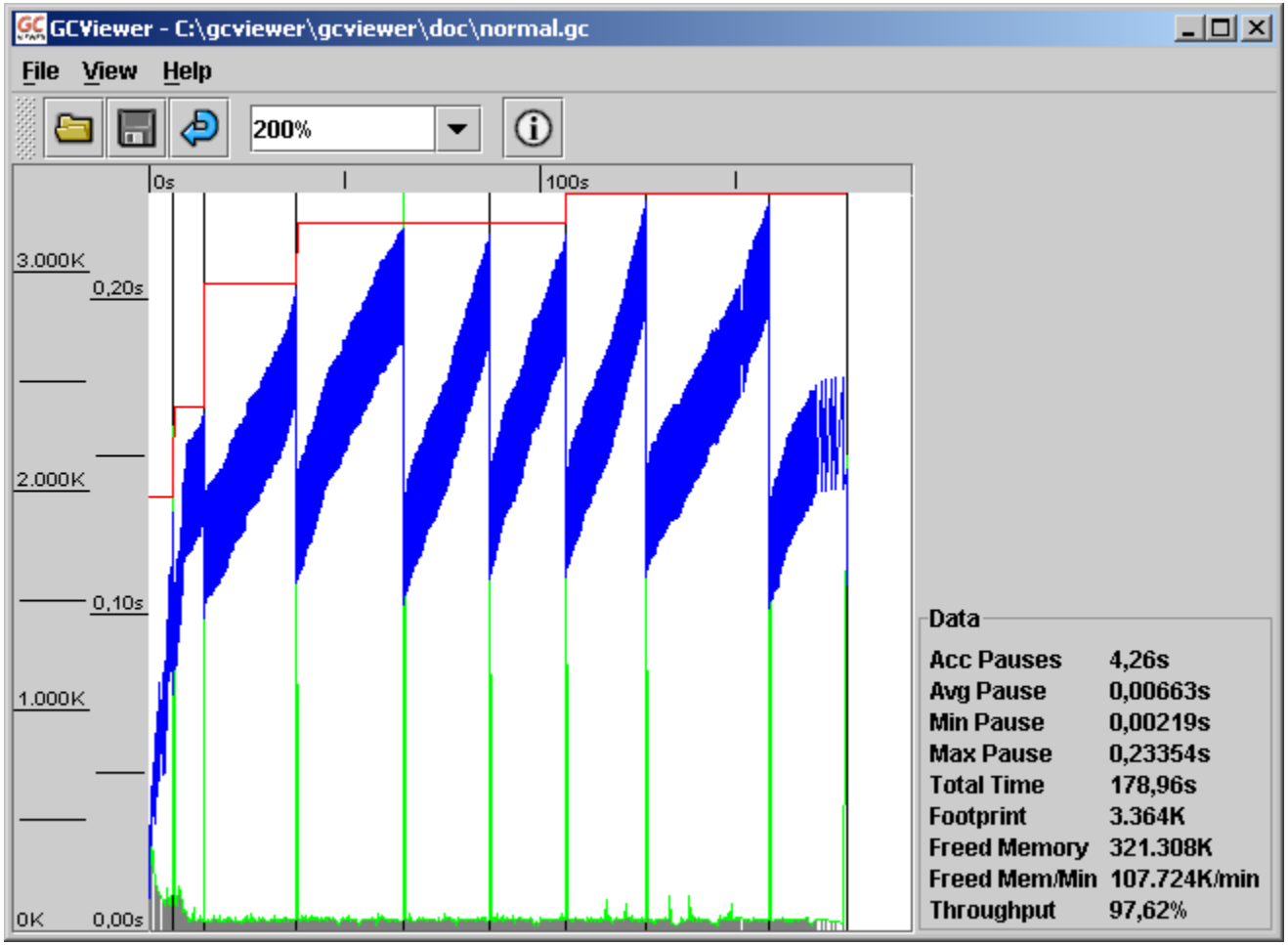
Weiterführende Elemente

- Konfigurationselemente, wie beispielsweise Authentifizierung, Cookies, Proxy Server, JDBC Connection Pools
- Pre- und Post-Processors
- Ant-Integration

5.4

DER GCVIEWER

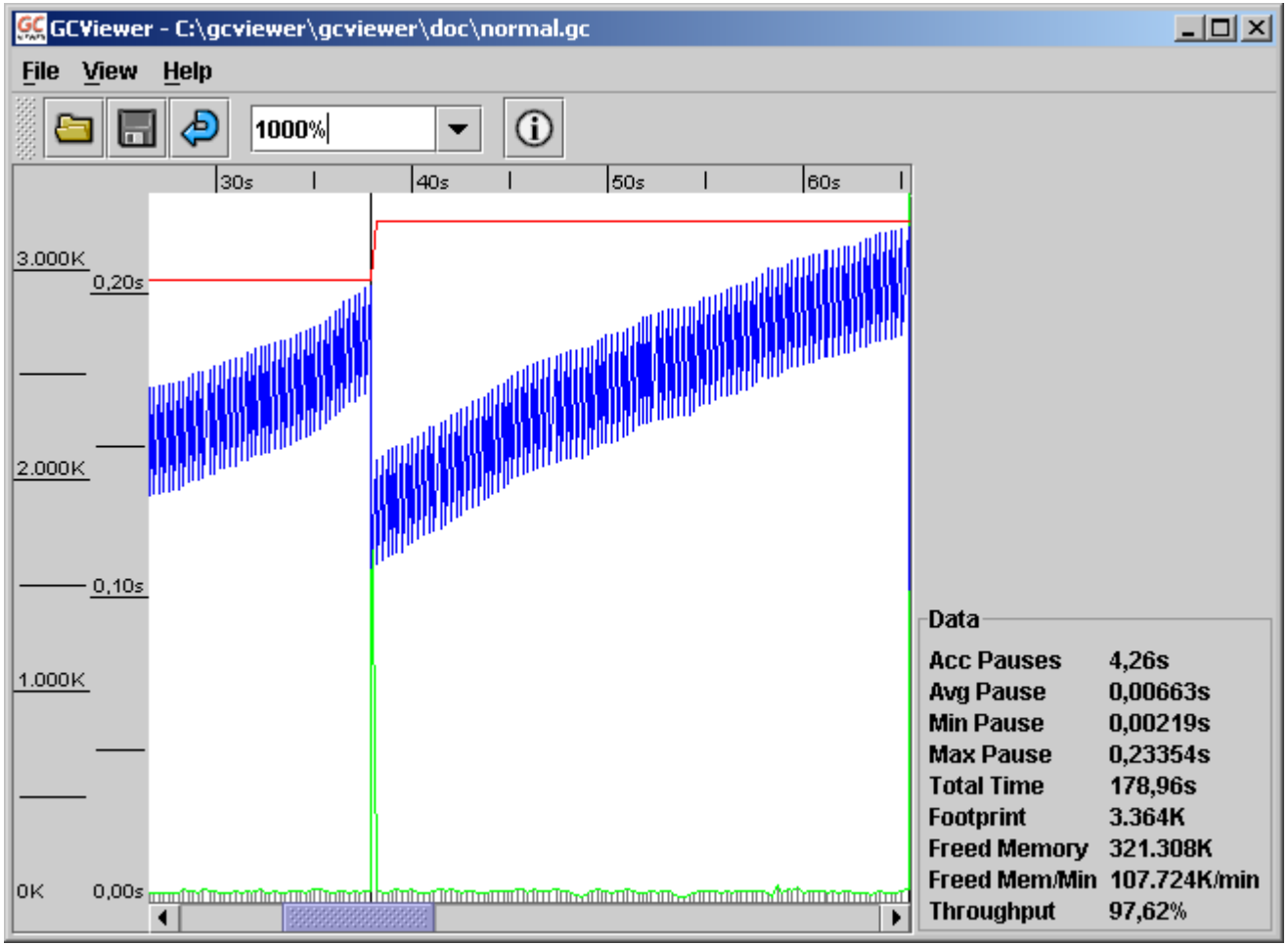
Oberfläche


GCViewer - C:\gcviewer\gcviewer\doc\normal.gc
File View Help
200%
0s
100s
3.000K
2.000K
1.000K
0K
0,20s
0,10s
0,00s
Data
Acc Pauses 4,26s
Avg Pause 0,00663s
Min Pause 0,00219s
Max Pause 0,23354s
Total Time 178,96s
Footprint 3.364K
Freed Memory 321.308K
Freed Mem/Min 107.724K/min
Throughput 97,62%

Beschreibung

- Die rote Linie stellt die aktuelle Größe des Heap-Speichers dar, die blaue Sägezahnlinie den aktuellen Speicherbedarf aller Objekte der Anwendung
- Die Sägezahnstruktur erklärt sich natürlich durch laufende Garbage Collections

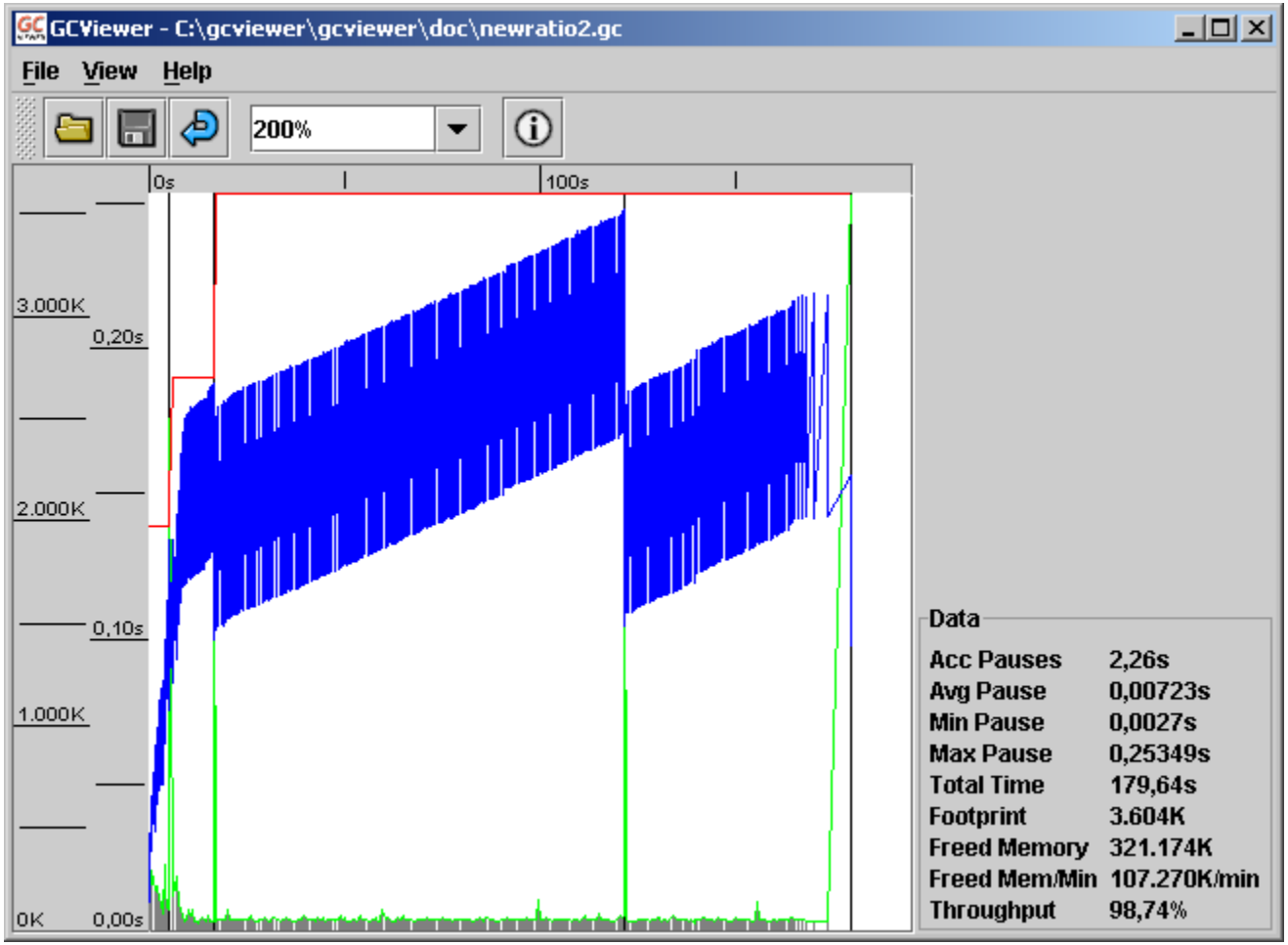
Minor Collections



Interpretation

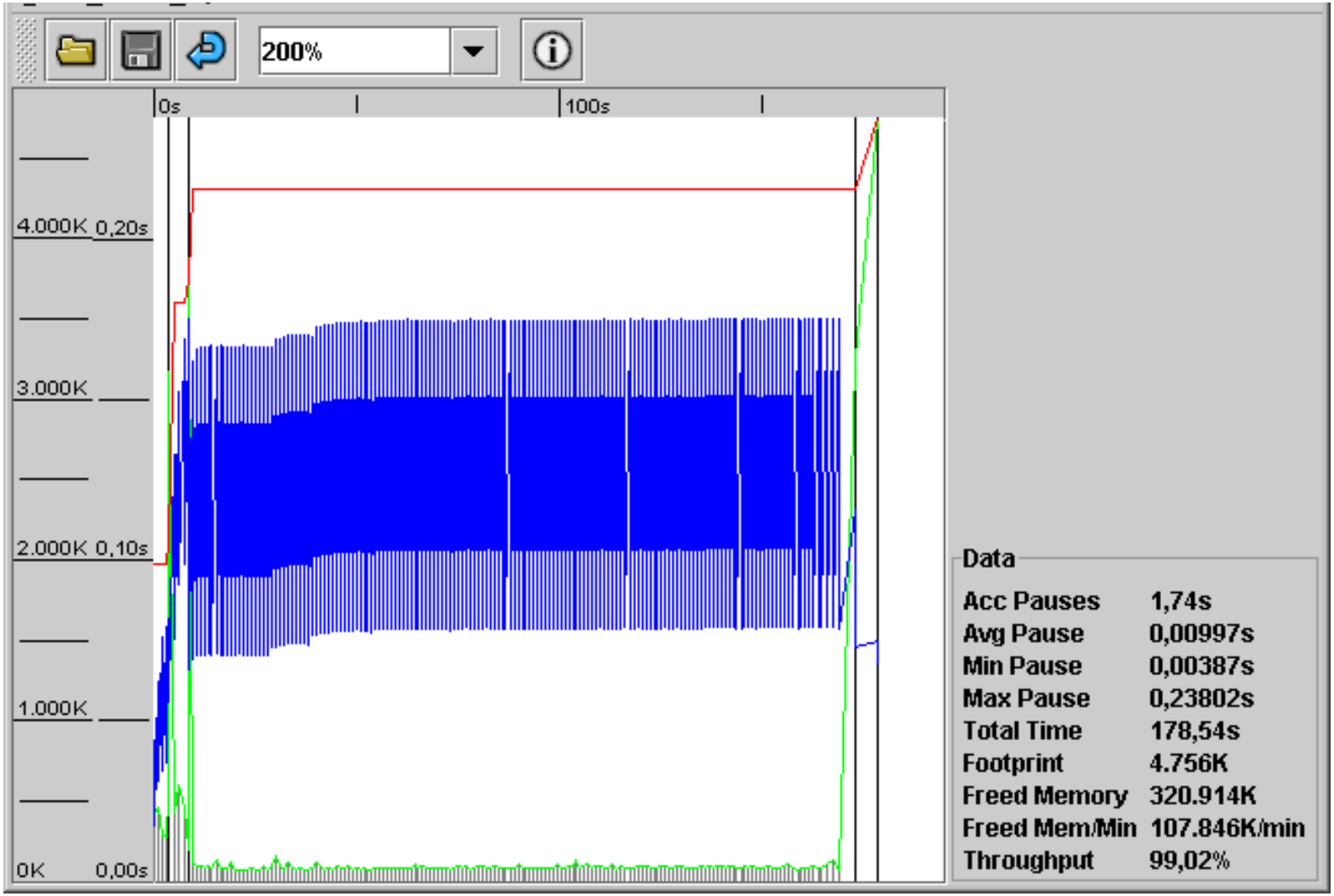
- Auch diese Zähne sind wiederum einzelne Garbage Collections
  - Diese Collections sind wesentlich häufiger und laufen auch deutlich schneller ab
  - Es sind die „Minor Collections“
- Die beiden Bilder zeigen nun in der Analyse, dass offensichtlich Objekte in die Tenured Generation wechseln, die anschließend von einer Major Collection entfernt werden müssen
  - Genau dies ist ein Zustand, der eventuell verbessert werden kann

Vergrößerung der New Generation



Optimierter Speicherverlauf

integrata cegos



integrata cegos

5.5

JMAP UND JHAT

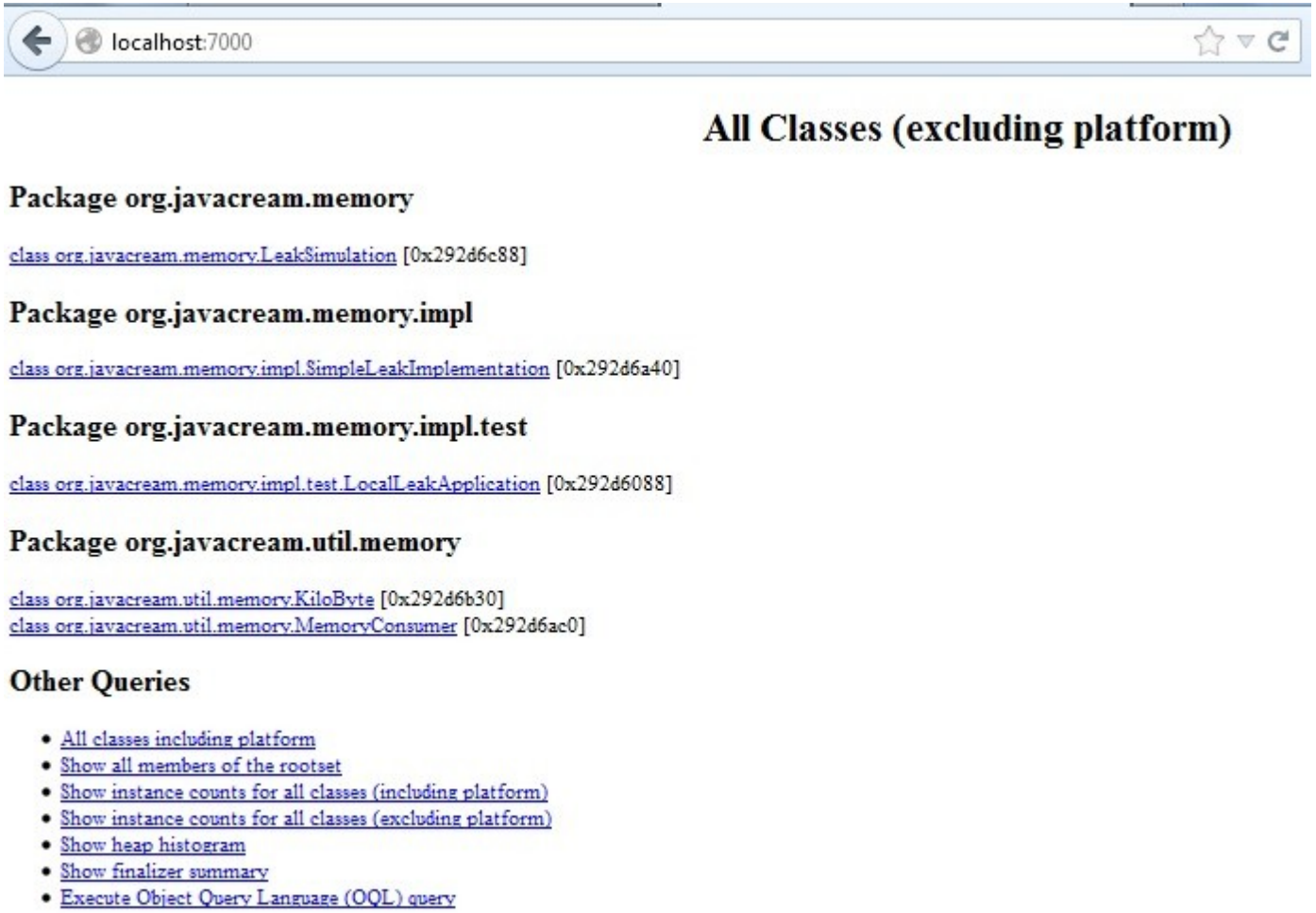
Beschreibung

integrata cegos

- jmap ist ein Werkzeug des JDK, das eine laufende Java Virtual Machine zum Schreiben einer Dump-Datei veranlasst
  - jmap –dump:format=b,file=dump.prof <pid>
- Dieser Dump kann anschließend durch ein Profiler-Werkzeug analysiert werden
  - Ein relativ einfacher Profiler steht mit jhat ebenfalls als Bestandteil des JDK zur Verfügung

jhat in Aktion

integrata cegos



integrata cegos

6

WEITERE BEGRIFFE UND TECHNOLOGIEN

6.1

TRANSAKTIONEN

- Atomar
  - Mehrere DBMS-Anweisungen müssen als Block ausgeführt werden
- Consistent
  - Nur bei Erfolg aller Aktionen, diese in die Datenbank schreiben (COMMIT)
  - Beim Scheitern mindestens einer Aktion muss die Datenbank wieder auf den Stand vor der Transaktion zurückgesetzt werden (ROLLBACK)
- Isolated
  - Grad der Isolation einer Transaktion von einer anderen
- Durable
  - Änderungen sind persistent gespeichert

Transaktionsarten

- Lokale Transaktion, Transaktion innerhalb eines DBMS
  - Es sind nur Tabellen einer Datenbank betroffen
- Verteilte Transaktion, Transaktion über mehrere DBMS hinweg
  - Es sind Transaktionen in mehreren Datenbanken betroffen

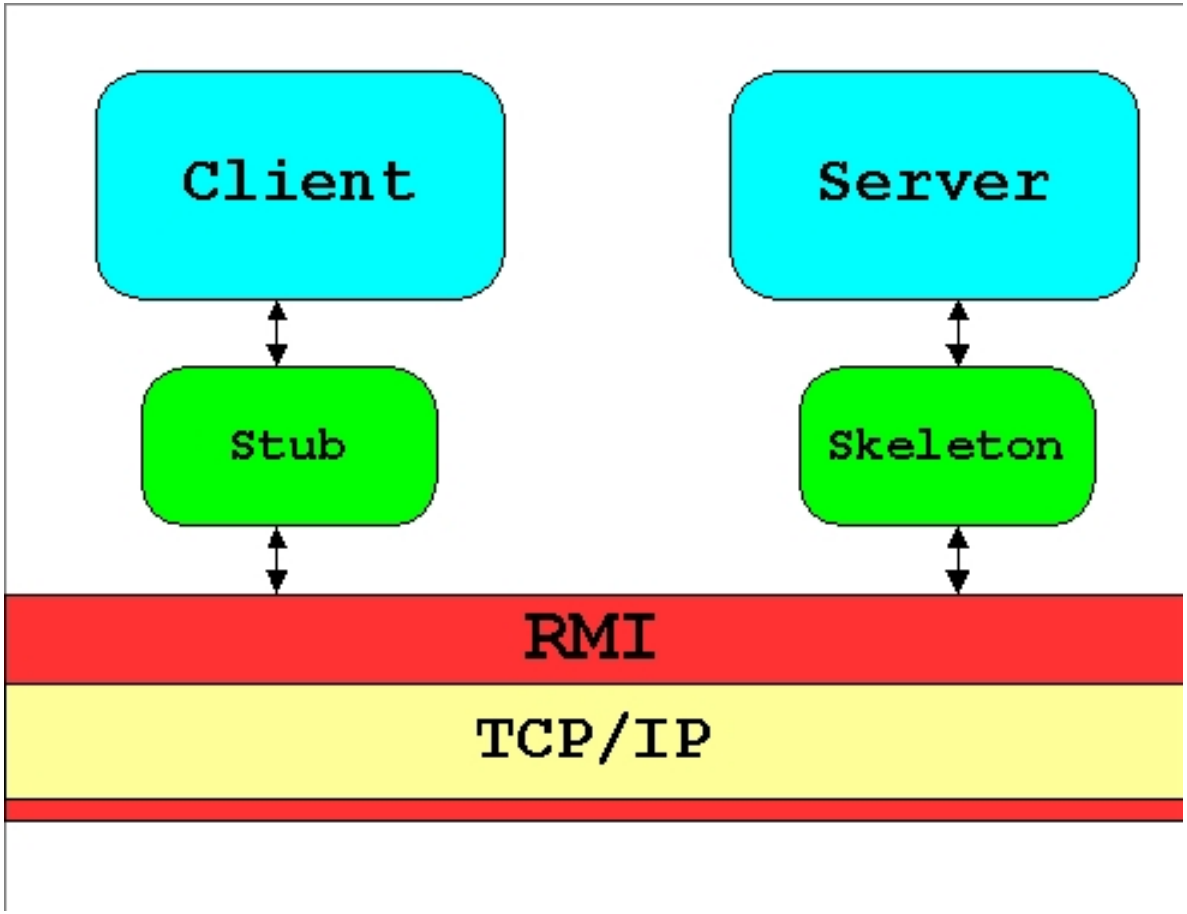
Isolation Level

- SQL definiert Transaktionsisolation in vier Graden durch drei Phänomene, die zwischen gleichzeitigen Transaktionen verhindert werden müssen
- Die unerwünschten Phänomene sind:
  - Dirty Read (Lesen ungültiger Daten)
    - Eine Transaktion liest Daten, die von einer gleichzeitigen, aber noch nicht abgeschlossenen Transaktion geschrieben wurden
  - Nonrepeatable Read (nichtwiederholbares Lesen)
    - Eine Transaktion liest Daten, welche sie vorher schon gelesen hat, erneut und stellt fest, dass die Daten von einer anderen Transaktion (die seit dem ersten Lesen abgeschlossen wurde) verändert worden sind
  - Phantom Read (Lesen von Phantomdaten)
    - Eine Transaktion führt eine Anfrage mit Suchbedingung, welche eine Gruppe von Zeilen ergibt, wiederholt aus und stellt fest, dass sich die Gruppe der Zeilen, die die Suchbedingung erfüllen, wegen einer anderen Transaktion, die seitdem abgeschlossen wurde, geändert hat

6.2

KOMMUNIKATIONSPROTOKOLLE

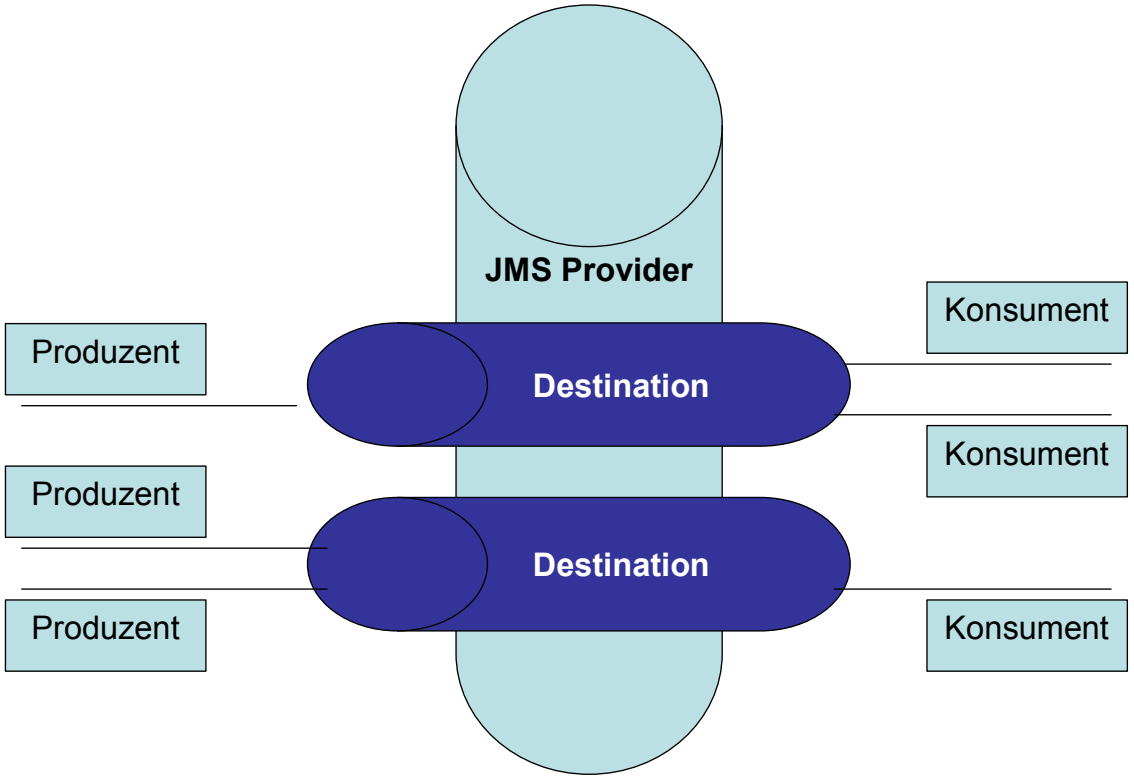
Remote Method Invocation



http

- Das Hypertext Transfer Protocol (http) erweitert das TCP/IP-Protokoll um mehrere standardisierte Elemente:
  - Der http-Header mit einem definierten Satz von Properties
  - Status-Codes für Server-Ergebnisse
    - So kennt wohl jeder Internet-Benutzer den Status-Code 404: „Not Found“
  - Daten-Typen und MIME-Types
- http verlangt keine stehende Verbindung zwischen Client und Server und kann damit zustandslos benutzt werden
  - Dies erleichtert den Umgang mit Firewalls deutlich
- http wird als Protokoll für alle Web-Anwendungen benutzt aber auch für die so genannten Web Services

Java Messaging Service JMS



JMS-Begriffe

integrata cegos

- Message
  - Eine Message besteht aus einem Header, einem Body mit Daten und einer Möglichkeit, den Empfang zu bestätigen
  - Der Header enthält beispielsweise das Ziel der Nachricht und eventuell eine Weiterleitungs- oder Antwortadresse
- MessageConsumer
  - Ein Konsument von Messages
- MessageProducer
  - Ein Produzent von Messages
- Topic
  - Ein Thema, zu dem der Produzent Nachrichten erzeugt, jeder interessierte Konsument wird vom Eintreffen einer Nachricht zu diesem Thema informiert
- Queue
  - Eine Message innerhalb einer Queue wird also von exakt einem Empfänger verarbeitet
- Message Selector
  - Ein Message-Selektor liefert einen logischen Ausdruck, der bei der Anmeldung eines Konsumenten beim Message-Provider hinterlegt werden kann

Web Services

- Web Services werden heute in zwei Ausprägungen betrieben:
  - SOAP-basierte Web Services tauschen zwischen Client und Server XML-Dokumente aus
    - Sie sind damit nicht direkt im Browser benutzbar
  - RESTful Web Services verzichten auf die zwangsläufige Benutzung von XML und benutzen stattdessen etablierte Dokumenten-Formate wie HTML, PDF und JSON

6.3

DIE KOMPONENTEN DER JAVA 2 ENTERPRISE EDITION

Servlets

integrata cegos

- Servlets sind
  - Ansprechbar über http
  - Zugriff über eine URL
- Der Container verwaltet häufig nur eine einzige Servlet-Instanz, die dann mit vielen verschiedenen Threads angesprochen wird
- Der Container ist für die Ausfallsicherheit der Komponente verantwortlich
- Servlets nehmen nur http-Aufrufe entgegen und interpretieren die darin enthaltenen Informationen als Methodenaufrufe
  - Das Servlet delegiert die Aufrufe weiter und enthält deshalb wenig bzw. gar keine Geschäftslogik

Arbeitsweise des Servlets

integrata cegos

- Lebenszyklus der Servlet-Instanz
  - Es genügt eine Servlet-Instanz pro Container
- Verwaltung des Zustands pro Request/Session
- Einfache Authentifizierung und Autorisierung
- Servlets müssen zustandslos und threadsicher konzipiert werden, so dass alle referenzierten Klassen müssen threadsicher sein müssen
- Die Http-Session kann zum Halten des Zustands verwendet werden; verlangt die Architektur Lastverteilung oder Ausfall-Sicherheit, ist Replizierung nötig

Enterprise JavaBeans

integrata cegos

- Entfernt ansprechbar über Java RMI und IIOP sowie JMS
  - Auch lokale Zugriffe innerhalb einer Virtuellen Maschine sind möglich
  - Zugriff über JNDI-Namen
- Der Container verwaltet den Lebenszyklus der Instanzen und legt häufig einen Pool gerade unbenutzter Objekte an
- Der Container ist für die Ausfallsicherheit der Komponente verantwortlich

Stateless SessionBeans

integrata cegos

- kein Client-typischer Zustand
- Ausfall-Sicherheit ist sehr gut, nicht abgefangen ist nur der Ausfall des Servers während eines Geschäfts-Prozesses
  - Falls die SessionBean jedoch „idempotent“ ist (mehrfache Aufrufe haben stets die gleichen Ergebnisse ohne Nebeneffekte), kann sogar in dieser extremen Situation neu aufgesetzt werden
- Einsatz-Kriterien
  - Lastverteilung,
  - Ausfallsicherheit,
  - Zustandslose Dienste

Stateful SessionBeans

integrata cegos

- Client-typischer Zustand wird auf Server-Seite gehalten
- Ausfallsicherheit befriedigend
- Passivierung/Aktivierung des Zustands erfolgt am Ende des Methodenaufrufs und am Ende einer Transaktion
  - Damit Probleme bei großen Sessions
- Einsatz-Kriterien
  - Server-Objekt enthält sinnvolle Zustands-Informationen (sinnvoll im Allgemeinen nur als Protocol-Adapter)
  - Vorsicht bei Verbindung von Stateful-Session-Beans (u.a. wegen evtl. unterschiedlicher Timeouts)

Stateless versus Stateful

integrata cegos

- Erfordert die Implementierung der Fachlogik das Halten des Zustands auf dem Server, bleiben nur Stateful SessionBeans übrig
- Die Performance der beiden Bean Typen ist bei einem einzigen Applika-tionsserver vergleichbar
- Im Cluster ist jedoch zu berücksichtigen, dass die Verwaltung von Stateful SessionBeans aufwändiger ist
  - Der Applikationsserver muss durch Replizieren des Conversational State die Ausfallsicherheit bzw. Lastverteilung gewährleisten
  - Auch hier ist jedoch zu berücksichtigen, dass diese Funktionalität notwendig ist und deshalb nicht aus Gründen der Geschwindigkeit geopfert werden kann

MessageDrivenBeans

integrata cegos

- Sind zustandslos
- Können Transaktionen starten, aber keinen Transaktions-Kontext propagieren
- Ausfallsicherheit:
  - Prinzipiell sehr gut, ein Problem ist gegebenenfalls nur die Ausfallsicherheit der Message-Destination
- Einsatz-Kriterien
  - Methoden ohne Rückgabewert
  - Höchste Ausfallsicherheit durch Zwischenspeicherung der Messages durch den Message-Provider vor der Ausführung der eventuell nicht einsatzbereiten Fachkomponenten

Connectors

- Ziel der Java Connector Architecture (JCA) ist die Integration eines vorhandenen Backend-Systems
  - Häufig ist dieses System transaktions-fähig und verlangt Authentifizierung
- Verwendung:
  - Werden über normale Methodenaufrufe angesprochen

Grenzen der JEE-Spezifikation

integrata cegos

- Grundlegende Anforderungen an eine stabile Middleware sind nicht immer spezifiziert, sondern nur implizit vorgesehen
  - Ein professioneller Applikationsserver wird diese Anforderungen selbstverständlich erfüllen
  - Die dafür notwendigen Konfigurationen sind jedoch leider Hersteller-abhängig
- Verfügbarkeit:
  - Fail-over
  - “Hot-Deployment”
  - Restart von Services
- Performance
  - Skalierbarkeit
  - Load-Balancing
  - Multi-Threading
  - Pooling
- System-Management

6.4

CLUSTERING

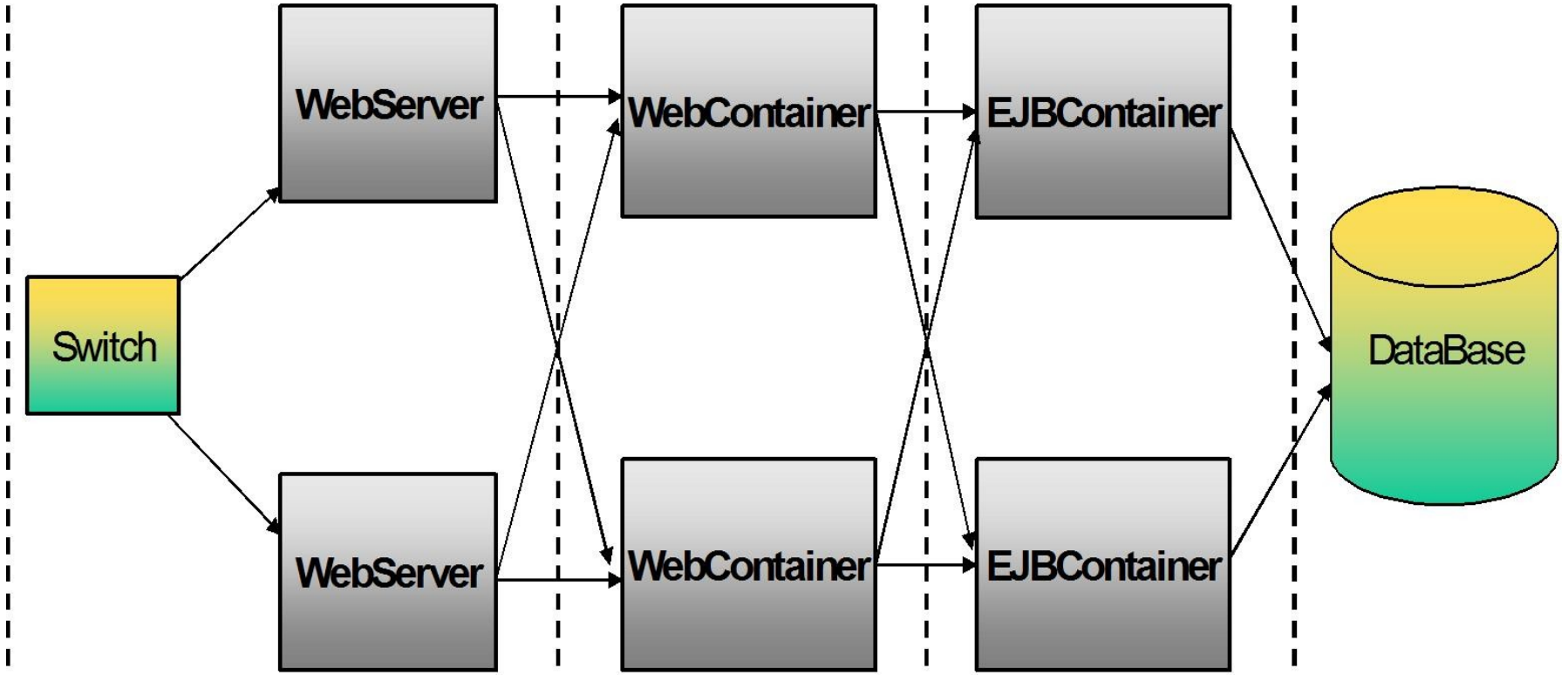
Notwendigkeit des Clusters

integrata cegos

- Als Beispiel für eine absolut notwendige Bedingung für eine geforderte 24x7-Verfügbarkeit des Applikationsservers im Hochlastbereich: Clustering
- Ein Cluster von Servern ist immer dann notwendig, wenn eines oder mehrere der folgenden Kriterien vorhanden sind:
  - Automatisches Fail-Over beim Ausfall eines Servers auf einen anderen Server
  - Kann das System mit steigender Last mitwachsen?
  - Ist ein Load-Balancing (Weitergabe von Anfragen an den am wenigsten belasteten Knoten) möglich?
  - Werden Anfragen auf einem Knoten parallel durchgeführt?
  - Pooling kritischer Ressourcen wie offene Datenbank-Verbindungen
  - Zentrale Management-Funktionen

Ein Cluster

integrata cegos



Beschreibung

- Ein Switch schaltet an Hand eines Algorithmus, z.B. einfaches Round robin, zwischen zwei Instanzen des Web Servers hin und her
- In obiger Abbildung sind nun ebenfalls zwei Applikationsserver vorhanden, die sich ebenfalls gegenseitig absichern
- Fällt einer aus, kann der andere beispielsweise Sessions übernehmen, wenn diese automatisch auf den jeweils anderen Server repliziert werden
- Das zentrale System bleibt das Enterprise Information System, hier eine Datenbank

Cluster mit Session-Verwaltung

integrata cegos

- Sollen Client-typische Sessions im Speicher des Servers abgelegt werden, darf der Load Balancer nicht beliebig zwischen Aufrufen zwischen Servern wechseln
  - Statt dessen: Sticky Sessions
- Problem: Woran erkennt der Load Balancer, welche Session er welchem Server zugeordnet hat?
  - Zuordnung des Clients über dessen IP-Adresse
    - Funktioniert nicht, falls Proxy-Server zwischen Client- und Server-Netzwerk liegen
  - Der Load-Balancer "weiß", wo im Request eine Session ID hinterlegt ist und analysiert diese für jeden Request
    - Fertige Lösungen existieren für http-Aufrufe

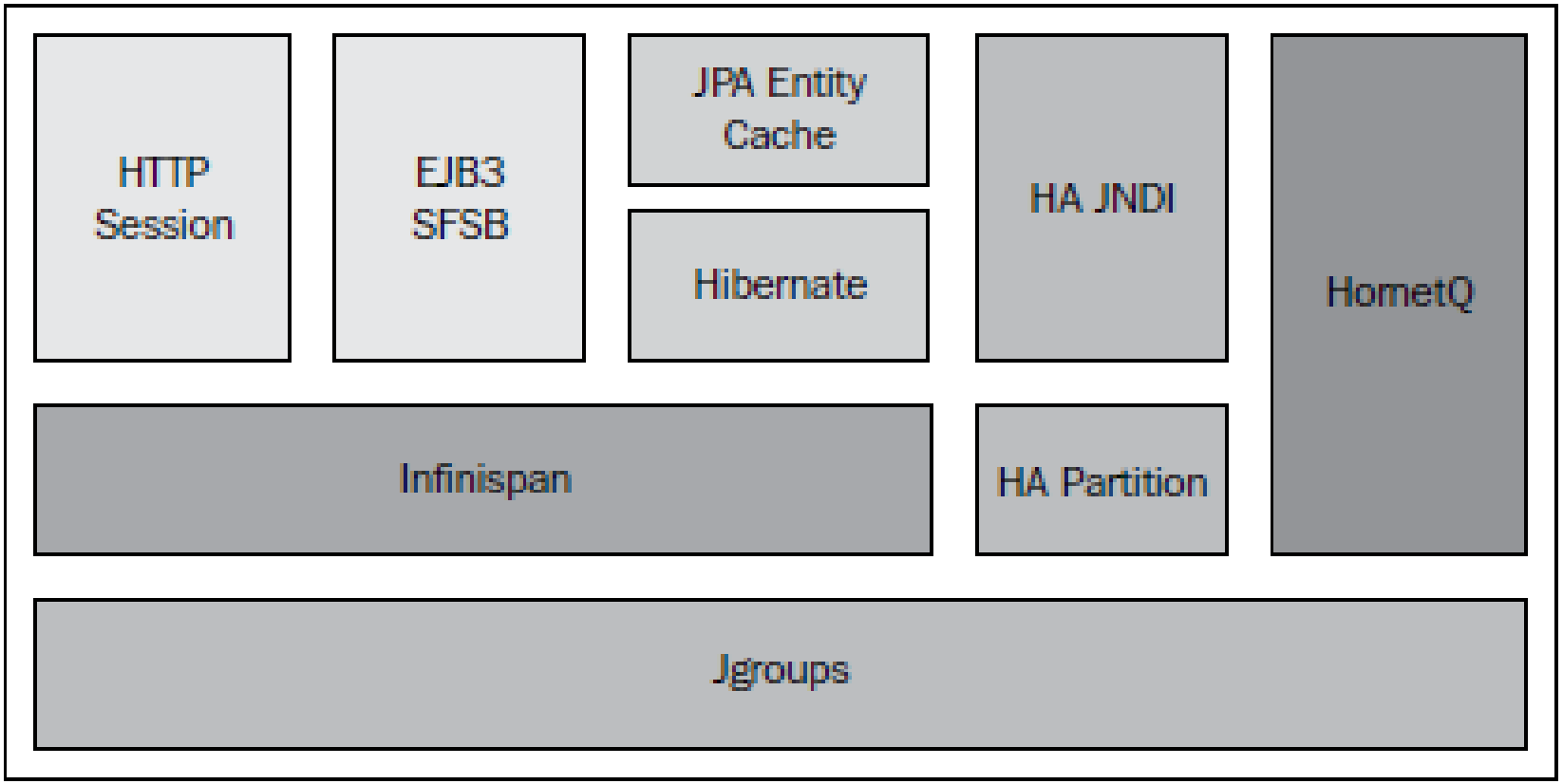
Cluster mit Session-Replikation

integrata cegos

- Müssen Sessions Ausfall-sicher verwaltet werden, müssen die Knoten des Clusters ihre Sessions replizieren
  - Zusätzlicher Aufwand, damit verschlechtert sich die Skalierbarkeit des Clusters
  - Die Knoten müssen nun miteinander kommunizieren können
- Im Ausfall-Fall muss der Server wissen, welcher andere Server die Session übernehmen kann
  - Session-Migration
  - Nicht trivial, da es meistens unmöglich ist, dass ein Knoten von allen anderen Knoten abgesichert wird
    - Statt dessen häufig "Replikationsgruppen"

Beispiel: Das Clustering-System von JBoss

integrata cegos



Copyright und Impressum

© Integrata Cegos GmbH

Integrata Cegos GmbH
Zettachring 4
70567 Stuttgart

Alle Rechte, einschließlich derjenigen des auszugsweisen Abdrucks, der fotomechanischen und elektronischen Wiedergabe vorbehalten.